



Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Final Project

**Adaptive Optics without wave-front
sensor using the APHEFIS (Adaptive
Optics by Phase Estimation from Focal
Intensity Speckles) algorithm**

**(Òptica Adaptativa basada en
l'estimació de la fase del camp al pla
focal, a partir de les estructures focals
d'intensitat)**

Studies: Enginyeria de Telecomunicació

Author: Manuel Muñoz Fuentes

Director: Dirk Giggenbach

Year: 2012

General Index

General Index	3
Acknowledgements	5
Abstract.....	7
Resumen del Proyecto.....	9
Resum del Projecte.....	11
1. Introduction	13
1.1. State of the art.....	13
1.2. Initial Algorithm Concept and Objectives	15
1.3. Structure of the Report	16
2. Theoretical basis	19
2.1. Mathematical tools.....	19
2.1.1. Fourier Transform	19
2.1.2. Phase unwrapping	21
2.1.3. Zernike Polynomials.....	22
2.2. Atmospheric Theory	24
2.2.1. General Theory	24
2.2.2. Aberrated Fields.....	27
2.2.3. Adaptive Optics.....	31
2.3. Other tools	34
2.3.1. Segmentation Tools: Watershed Algorithm.....	34
2.3.2. Artificial Neural Networks	37
3. Phase Estimation from Focal Intensity Speckles algorithm.....	41
3.1. Introduction.....	41
3.2. Test Environment	43

3.3.	Considerations regarding the symmetries of the phase.....	47
3.4.	Algorithm assumptions	51
3.4.1.	Two levels evaluation of the phase	51
3.4.2.	Division in constant phase zones	54
3.4.3.	Treatment of only High Power Zones.....	56
3.4.4.	Numerical analysis of the assumptions	57
3.5.	Implementation of the Phase Prediction	59
3.5.1.	Directly observable Relations.....	59
3.5.2.	Machine Learning Relations.....	62
4.	Conclusions and Future Lines of Work	69
5.	Appendix	73
	Appendix A: Acronyms	73
	Appendix B: Files index	74
	Created	74
	Adapted or not modified:.....	76
	Appendix C: GUI guide.....	78
6.	References.....	89

Acknowledgements

I want to thank the DLR for giving me the opportunity of making this project in the Oberpfaffenhofen installations. It has been one of the most filling experiences in my life. In special, I wish to sincerely thank two persons. First to Dirk Giggenbach, the creator of the algorithm used and analyzed and the main brain in the project. And second, but not less important, to Ramon Mata, my main contact in the institution, and the person who has helped me more during my stay, both in the realization of the project and in solving the unavoidable difficulties of working in a foreign environment.

Abstract

With the radio channels reaching its limits in Satellite communications, the optical links are each day a more attractive solution to transmit the huge amounts of data that the sensors produce nowadays. However, to be able do so it is necessary to overcome the distortions introduced by the atmosphere, which become much worse at optical frequencies. This is what creates the need of an Adaptive Optics (AO) system for such communications.

Originally, the AO systems where developed for astronomy, where contact times are much larger. Communications with Low Earth Orbit (LEO) satellites, for example, have short contact times and needs therefore to exploit the link until very low elevation angles. Furthermore the ground stations for communications will normally be at lower altitude, in front of the astronomic installations in regions with optimum atmospheric characteristics. In such conditions the turbulence impact is much higher than in astronomy. Another main difference is the slew-rate of LEO satellites. Crossing rapidly a big amount of atmosphere volume makes the turbulence fluctuations change faster. Therefore the AO bandwidth required to compensate the phase distortion is higher in communications scenarios than in astronomy. Both strong turbulence and high AO bandwidth have to be taken into account in the development of an AO system.

The AO system corrects the phase distortions by changing the shape of a deformable mirror to fit the incoming wave-front. The wave-front sensor estimates the phase and sends the information to the control computer, which closes the correction loop. The AO system, and the wave-front sensor in particular, is generally expensive, not easily operated or moved, and very sensible to calibration errors.

A new AO technique, developed by Dr. Giggenbach [GIG10][GI11a], makes profit of the properties of the optical receiver system to make a prediction of the phase. Concretely it uses the information provided by the focus camera and assumes a correlation between phase and amplitude of the optical field in the focal plane. This

correlation in the focal plane allows estimating the wave-front phase in the aperture with little hardware complexity.

In this document, the theoretical bases related with this algorithm and with atmospheric optical communications are exposed, together with other tools used for implementing of the algorithm.

Afterwards, the technique is exposed in more detail. The concepts and simplifications are analyzed and evaluated individually. For this scope, an AO simulation environment has been developed and tested with simulated complex fields.

Later, several implementations have been analyzed. After the direct implementations have failed to produce the desired performance, a machine learning algorithm has been developed. This system has been tested and evaluated, obtaining the results exposed in the corresponding chapter.

Finally, conclusions regarding the feasibility and effectiveness of a system based in the APHEFIS algorithm have been extracted, and the most interesting lines of work for future research have been stated.

Resumen del Proyecto

Con los canales de radio llegando a su límite en comunicaciones por satélite, los enlaces ópticos son cada día una solución más atractiva para transmitir las enormes cantidades de datos que los sensores producen. Sin embargo, para poder hacerlo es necesario superar las distorsiones introducidas por la atmósfera, mucho peores con tecnologías ópticas. Esto crea la necesidad de una Óptica Adaptativa (AO) para tales comunicaciones.

Originalmente, los sistemas AO fueron desarrollados para astronomía, donde los tiempos de contacto son mucho mayores. En comunicaciones por satélite, sobretudo en Baja Órbita Terrestre (LEO), los tiempos de contacto en cambio son mucho más cortos y por tanto se debe aprovechar la conexión hasta elevaciones muy bajas. Además, las estaciones terrestres para comunicaciones estarán normalmente a bajas altitudes, frente de las instalaciones astronómicas situadas en regiones con óptimas características atmosféricas. En esas condiciones, el impacto de las turbulencias es mucho mayor. Otra diferencia importante es la gran velocidad de los satélites LEO. Cruzando a gran velocidad un importante volumen atmosférico, las fluctuaciones en las turbulencias cambian mucho más rápido. Por tanto, en sistemas AO usados en comunicaciones el ancho de banda requerido para compensar estas distorsiones es mucho mayor que en astronomía. Tanto el ancho de banda como las fuertes turbulencias han de ser tenidos en cuenta en el desarrollo de un sistema AO.

El sistema AO corrige las distorsiones de fase cambiando la forma de un espejo deformable para adaptarse al frente de onda recibido. El sensor de frente de onda calcula la fase y envía la información al ordenador de control, que cierra el lazo de corrección. El sistema AO, y el sensor de frente de onda en particular, es generalmente es caro, difícil de operar y trasladar, y muy sensible a los errores de calibración.

Una nueva técnica AO, desarrollada por el Dr. Giggenbach [GIG10] [GI11a], se beneficia de las propiedades del sistema receptor óptico para hacer una predicción

de la fase. Concretamente, utiliza la información proporcionada por la cámara de rastreo y asume una correlación entre fase y la amplitud del campo óptico en el plano focal. Esta correlación en el plano focal permite estimar la fase de frente de onda en la abertura con una baja complejidad de hardware.

En este documento, las bases teóricas relacionadas con este algoritmo y con las comunicaciones ópticas atmosféricas son expuestas, en conjunto con otras teorías relacionadas con la implementación del algoritmo.

Después de eso, la técnica se expone con más detalle. Cada uno de los conceptos y las simplificaciones son analizados y evaluados individualmente. Para ello, un entorno que simula el comportamiento de un sistema AO ha sido desarrollado y probado con campos generados artificialmente.

Posteriormente, diferentes implementaciones se han analizado. Después de que implementaciones directas fallasen en producir la eficiencia deseada, un algoritmo de machine learning ha sido desarrollado. Este sistema ha sido probado y evaluado.

Por último, conclusiones en relación con la viabilidad y la eficacia de un sistema basado en el algoritmo APHEFIS han sido extraídas, y las líneas de trabajo más interesantes para futuras investigaciones se han establecido.

Resum del Projecte

Amb els canals de ràdio arribant al seu límit en comunicacions per satèl·lit, els enllaços òptics són cada dia una solució més atractiva per transmetre les enormes quantitats de dades que els sensors produeixen. No obstant això, per fer-ho és necessari superar les distorsions introduïdes per l'atmosfera, molt pitjors amb tecnologies òptiques. Això crea la necessitat d'una Òptica Adaptativa (AO) per a tals comunicacions.

Originalment, els sistemes AO van ser desenvolupats per astronomia, on els temps de contacte són molt més grans. En comunicacions per satèl·lit, sobretot en Baixa Òrbita Terrestre (LEO), els temps de contacte en canvi són molt més curts i per tant s'ha d'aprofitar la connexió fins elevacions molt baixes. A més, les estacions terrestres per a comunicacions estaran normalment a baixes altituds, davant de les instal·lacions astronòmiques situades en regions amb òptimes característiques atmosfèriques. En aquestes condicions, l'impacte de les turbulències és molt més gran. Una altra diferència important és la gran velocitat dels satèl·lits LEO. Creuant a gran velocitat un important volum atmosfèric, les fluctuacions en les turbulències canvien molt més ràpid. Per tant, en sistemes AO usats en comunicacions l'ample de banda requerit per compensar aquestes distorsions és molt major que en astronomia. Tant l'ample de banda com les fortes turbulències han de ser tinguts en compte en el desenvolupament d'un sistema AO.

El sistema AO corregeix les distorsions de fase canviant la forma d'un mirall deformable per adaptar-se al front d'ona rebut. El sensor de front d'ona calcula la fase i envia la informació a l'ordinador de control, que tanca el llaç de correcció. El sistema AO, i el sensor de front d'ona en particular, és generalment és car, difícil d'operar i traslladar, i molt sensible als errors de calibratge.

Una nova tècnica AO, desenvolupada pel Dr Giggenbach [GIG10] [GI11a], es beneficia de les propietats del sistema receptor òptic per fer una predicció de la fase. Concretament, utilitza la informació proporcionada per la càmera de rastreig i assumeix una correlació entre fase i l'amplitud del camp òptic en el pla focal.

Aquesta correlació en el pla focal permet estimar la fase de front d'ona en l'obertura amb una baixa complexitat de hardware.

En aquest document, les bases teòriques relacionades amb aquest algorisme i amb les comunicacions òptiques atmosfèriques són exposades, en conjunt amb altres teories relacionades amb la implementació de l'algorisme.

Després d'això, la tècnica s'exposa amb més detall. Cadascun dels conceptes i les simplificacions són analitzats i avaluats individualment. Per a això, un entorn que simula el comportament d'un sistema AO ha estat desenvolupat i provat amb camps generats artificialment.

Posteriorment, diferents implementacions s'han analitzat. Després que implementacions directes fallessin a produir l'eficiència desitjada, un algoritme de machine learning ha estat desenvolupat. Aquest sistema ha estat provat i avaluat.

Finalment, conclusions en relació amb la viabilitat i l'eficàcia d'un sistema basat en l'algoritme APHEFIS han estat extretes, i les línies de treball més interessants per a futures investigacions s'han establert.

1. Introduction

1.1. State of the art

One of the most important parts in satellite communications is the downlink to the Earth. Any information generated by any of the other parts becomes useless if it cannot be sent, and because of this the downlink capacity is one of the main constraints when designing it.

Nowadays, radiofrequency (RF) is mostly used for satellite communications. But as the amount of data to transmit increases, RF transmissions will soon reach its limit. This creates a bottle-neck that in great measure conditions the amount of information that can be obtained from a satellite. For example, a high resolution camera generates 6,7Gbit/s. A typical RF downlink has a data rate up to 256Mbit/s and can have a daily contact time up to 2360s, resulting in a total transferable data of around 75 Gb/day. This means that only 0.1% of the total possible information will be transferred [GIG07].

One of the most interesting alternatives is to use free space optic links to the satellite-to-ground communication links (SGL). Due to its good characteristics which will be discussed more in detail in the next chapter, the optical communications are one of the most promising technologies in SGL. They are already used for inter-satellite communications, achieving data rates higher than 5,5Gbps [LAN05], and they expect to improve even higher with the technologies developed in the last years.

Because of this, investigation towards this direction is currently being made by different companies and institutes. Coherent and non-coherent downlinks have been already carried out (i.e. with the OICETS and TerraSAR-X satellites [PER08] [GIG09]), showing the performance of free-space optical communications for SGL is incredibly high.

However, in the satellite to earth communication there is a big portion of the atmosphere between the transmitter and the receiver. The atmosphere has turbulences, which causes an index of refraction variation. This variation makes the signal not uniform, creating distortions in the received field which prevent the direct coupling in a single mode fiber, as would be desirable. Because of this, the use of AOs in order to correct the distortions of the atmospheric turbulence is indispensable.

The AO systems to compensate the atmosphere distortion of the signal are not new, and in fact are used in Astronomy since a long time ago. But most of the previous systems are less constrained in time. Also in Astronomy the images are made by looking always at the same point in the sky, which implies that the atmosphere will not change so fast. The SGL of LEO satellites is only available for times shorter than 10 minutes, so all the trajectory of the satellite must be used in order to be able to download the maximum data possible. Both the faster changes in the atmosphere and the higher distortions imply that a perfectly valid AO system for Astronomy will need in general considerable changes to be used in SGL.

Currently, one of the most important elements of an AO system is the wave-front sensor. This sensor is used to detect the phase distortions present in the aperture plane of the receiver. This information is passed to a computer which processes it, and afterwards uses it to pass the orders to a deformable mirror. This deformable mirror corrects the input field, the wave-front sensor measures the distortions again, and the loop starts again. Unluckily, the determination of the phase is a very difficult task. Working with low incidence angles does not only imply higher distortions, but also the appearance of singularities (named branchpoints) which can greatly increase the complexity of the detection.

Moreover, the incorporation of this wave-front sensor greatly increases the complexity of the system. It is a highly complex hardware which needs a very accurate calibration for each scenario, making it expensive and little adapted for mobile applications. However, the measurement or estimation of the phase is an unavoidable step in the correction of the signal, and nowadays they are the best option.

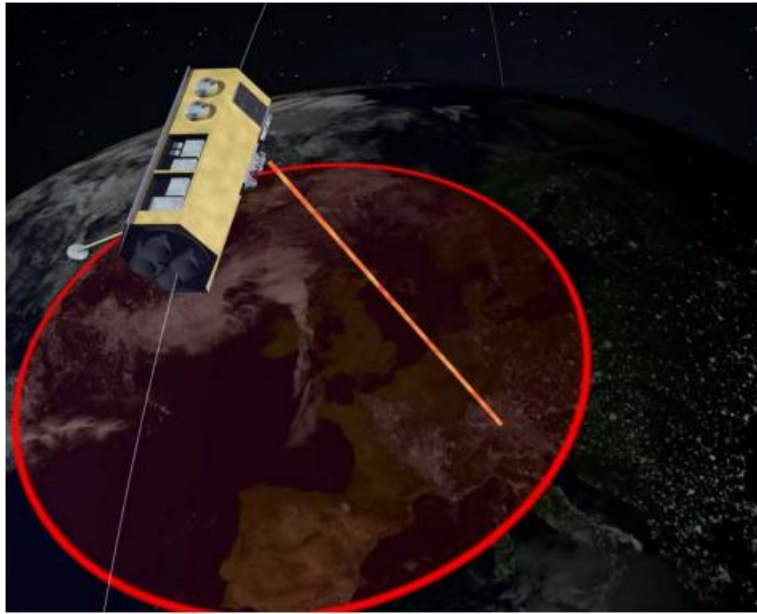


Figure 1.1: Downlink scenario from TerraSAR-X

1.2. Initial Algorithm Concept and Objectives

The objective behind the APHEFIS algorithm of Dr. Dirk Giggenbach [GIG01][GIG02] is to minimize the impact of the atmosphere while simplifying the actual standard system. The algorithm eliminates the need of a wave-front sensor, greatly improving the hardware simplicity of the system.

The estimation of the phase is obtained by a software treatment of the information that can be directly extracted from the field in the focus, taking advantage of the focusing. The algorithm involves Fourier operations and image processing tools. With this, the hardware complexity is highly reduced while the software one is increased. However, it is normally easier to deal with this one. The algorithm will be later discussed in detail in chapter 4.1.

The main objectives of this work are:

- To create a function structure in Matlab for extracting all the necessary data and treating the signal. An initial script developed by Dr. Giggenbach was provided with some of the basic functions.
- To determine different ways to treat the signal following the restrains of the algorithm. This treatment should input a distorted field and output the corrections that should be made.
- To determine a metric to evaluate the quality of each of the ways of treating the signal with the algorithm proportionated, and use it to determine the best way to do it.
- To analyze the limitations of the algorithm and future possible lines of work related with it.

1.3. Structure of the Report

In chapter two, for a better understanding of the subject, an explanation of the theoretical bases behind the Atmospheric Optics and Optics in general will be presented. After that, there will be a more general explanation of the other topics related with the subject, as the Fourier Transform, some image processing tools, and the behavior of a neural network.

In chapter three, with all the theoretical bases already known, the algorithm will be exposed and a dedicated analysis of it will be made. The different assumptions in the general algorithm concept will be evaluated, and the effect that each of them have in the quality of the correction will be determined.

Later on chapter four, the theory of the previous chapter will be used to present the different lines of work for the implementation of the algorithm that have been followed during the development of the project. After that, the most promising ones further explained. They will be compared and evaluated, and there will be stated which of them got better results.

After that, in chapter five, different implementations will be analyzed. After the direct implementations have failed to produce the desired efficiency, a machine learning algorithm has been adapted for its use in this ambit, and it has been used for obtaining a more accurate system. This system has been tested and evaluated, obtaining the results exposed in the corresponding chapter.

Afterwards, in chapter six some more insight on the theory behind the algorithm will be presented. The characteristics that seem to have a bigger impact in the behavior of the system, like the symmetries, will be presented and analyzed in more detail.

Last, with all the information obtained from the previous chapters, will be stated the conclusions that can be stated of all the process. Besides, the most promising future lines of work regarding the topic will be stated.

2. Theoretical basis

2.1. Mathematical tools

2.1.1. *Fourier Transform*

The Fourier Transform is one of the most used functions in communications. Its basic expression in (2.1.1) and it is valid with little changes in many fields. Regarding optical communications, it is the expression which describes the effect of focalization under normal circumstances.

$$\mathcal{F}\{g(x, y)\} = \iint_{-\infty}^{\infty} g(x, y) e^{-j2\pi(xf_x + yf_y)} dx dy$$

Eq. 2.1.1: Fourier Transform Expression

$$\mathcal{F}\{g(x, y)\} = \frac{1}{NM} \sum_{x=1}^N \sum_{y=1}^M g(x, y) e^{-j2\pi\left(\frac{xu}{N} + \frac{yv}{M}\right)}$$

Eq. 2.1.2: Discrete Fourier Transform

The Discrete Fourier Transform (DFT) (3.2) is a variation implemented for the necessity of dealing with discrete values. The continuous version is very useful for extracting theoretical conclusions, but in most cases is not practical because when processing the values are always discrete. Furthermore, there are very fast algorithms to calculate the DFT, as the Fast Fourier Transform (FFT), while the Fourier Transform is very complex to calculate in general cases.

One of its most important characteristics is its relation with the Fourier series, which lets us express any analytical function as a sum of sinus and cosines or complex exponentials. This property has many uses: from data compression (there is only need to store a couple of coefficients, instead of a full function), to the observation of frequency domain properties of the function. A function with most of its power concentrated in the low components will be a smooth function with no discontinuities or punctual variations, while a function with high components will be more abrupt in some points

It is generally used to represent a time function in its frequency domain. However, in this work it will be used to represent a space to frequency transform, because this is the operation done by a circular lens focusing a field.

The characteristics of the Fourier Transform more used in this project are:

- **Linearity:**

$$\mathcal{F}\{ag(x, y) + bh(x, y)\} = a\mathcal{F}\{g(x, y)\} + b\mathcal{F}\{h(x, y)\} \quad (\text{Eq 3.3})$$

- **Shift:**

$$\mathcal{F}\{g(x - x_0, y - y_0)\} = \mathcal{F}\{g(x, y)\}e^{-j2\pi(f_x x_0 + f_y y_0)} \quad (\text{Eq 3.4})$$

- **Parseval Theorem**

$$\iint_{-\infty}^{\infty} |g(x, y)|^2 dx dy = \iint_{-\infty}^{\infty} |\mathcal{F}\{g(x, y)\}|^2 df_x df_y \quad (\text{Eq 3.5})$$

- **Inverse**

$$\mathcal{F}^{-1}\{g(x, y)\} = \iint_{-\infty}^{\infty} g(x, y)e^{j2\pi(xf_x + yf_y)} dx dy \quad (\text{Eq 3.6})$$

- **Symmetries:**

$$\text{If } f_e(x, y) = \text{Even}(f(x, y)), \text{ and } f_o(x, y) = \text{Odd}(f(x, y)), \quad (\text{Eq 3.7})$$

$$\mathcal{F}\{f(x, y)\} = \iint_{-\infty}^{\infty} f_e \cos(2\pi(xf_x + yf_y)) dx dy - j \iint_{-\infty}^{\infty} f_o \sin(2\pi(xf_x + yf_y)) dx dy$$

$$\text{If } g(x, y) \text{ Real, then } \mathcal{F}\{g(x, y)\} = \mathcal{F}\{g^*(-x, -y)\} \quad (\text{Eq 3.8})$$

$$\text{If } g(x, y) \text{ Imag. then } \mathcal{F}\{g(x, y)\} = -\mathcal{F}\{g^*(-x, -y)\} \quad (\text{Eq 3.9})$$

2.1.2. Phase unwrapping

It must be noted that the optical field is a **complex** field, which means it has amplitude and phase. Or, what is the same, a real part and an imaginary part. As an easy way to visualize it, we can see it as two sinusoids with a displacement of $\pi/4$ between them, which advance both in time and space (Figure 2.1.1).

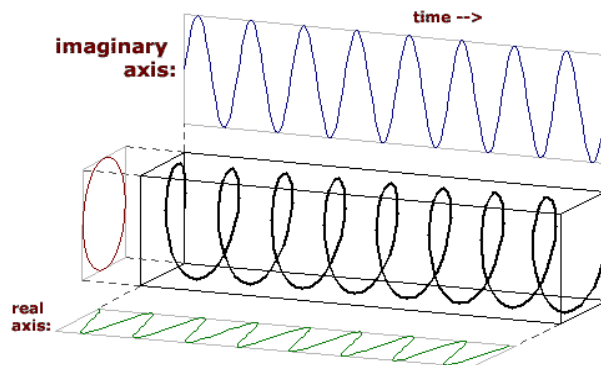


Figure 2.1.1: Visualization of a complex wave

It is well known that the phase of a complex number is 2π periodic with respect to its phase, so the most common representation of it is a wrapped version of it, which restricts all the values between $-\pi$ and π , or between 0 and 2π .

For most applications this is enough and even favorable, because it's easier to see on naked eye the quadrant in which the signal falls and many other properties. But working with this method becomes impossible when there is need to work with continuous signals, because it introduces many discontinuities. For this purposes, we need an unwrapped version of the phase (Figure 2.1.2).

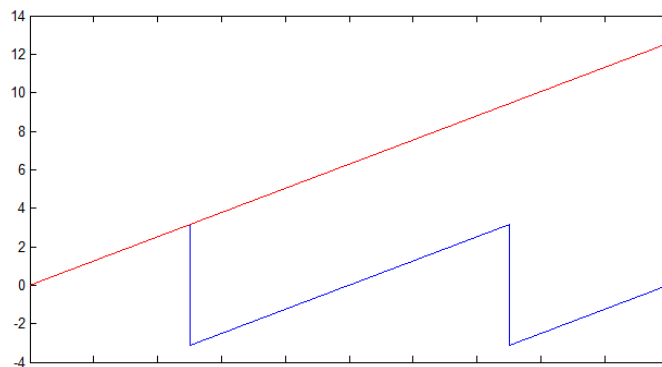


Figure 2.1.2: Two cycles of a wrapped (blue) and unwrapped (red) continuously increasing phase

This process of unwrapping in 1D signals is trivial, but in 2D it becomes more complex and computationally requiring, and not always will be possible to do it.

In particular, there will be some points named “branchpoints” in which this will be impossible by definition. These points are present where the amplitude of the signal goes to zero, and are characterized by the discontinuities they introduce in the phase of the signal. They will be further explained in Chapter 2.2.2.

Unwrapping the phase is crucial for working with many operators, like, for example, to decompose the signal in Zernike polynomials.

2.1.3. Zernike Polynomials

Any optical aberration of a complex field through a circular support area can be expressed as an infinite pondered summation of polynomials, called the Zernike polynomials. The coefficients multiplying these polynomials have the weight each kind of aberration has in the field. Generally, the lower orders have most of the power, so the coefficients decrease while the order increases. It is also possible to see how distorted a phase is by looking how fast this decrease goes.

The Zernike polynomials are a particular complete set of orthonormal polynomials (The requirement for having a valid base) which are a product of a radial function $R_n^m(\rho)$ and an angular function $G^m(\theta)$, where ρ is the normalized radial distance, θ the azimuthal angle, and m (also known as the frequency) and n (also known as the order) are nonnegative integers with $n \geq m$. These functions are defined as follows (Equation 2.1.3) for $n-m$ even:

$$R_n^m(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s (n-s)!}{s! [0.5(n+|m|)-s]! [0.5(n-|m|)-s]!} \rho^{n-2s}$$

$$G^m(\theta) = \begin{cases} \cos(m\theta) & \text{for } m \geq 0 \\ \sin(m\theta) & \text{for } m < 0 \end{cases}$$

Equation 2.1.3: Zernike Polynomials

For $n-m$ odd, the coefficients are zero. This means that for even orders, we only have even frequencies, and for odd orders we only have odd frequencies. For an easiest visualization, we can see the representation of the five first orders in (Figure 2.1.3).

A field with only the low orders of Zernike components, the tip and tilt, is a very good one. They are in fact only an angular displacement, and they can be easily measured and corrected with methods highly extended nowadays. High order components, however, represent a high challenge even for the most modern systems. It is important to clarify that mathematically sometimes the term “order” is used to describe each polynomial instead of each level of them, going up to down and left to right. So the Z_0^0 will be the Z_1 , the Z_1^{-1} will be the Z_2 , the Z_1^1 will be the Z_3 , and so on.

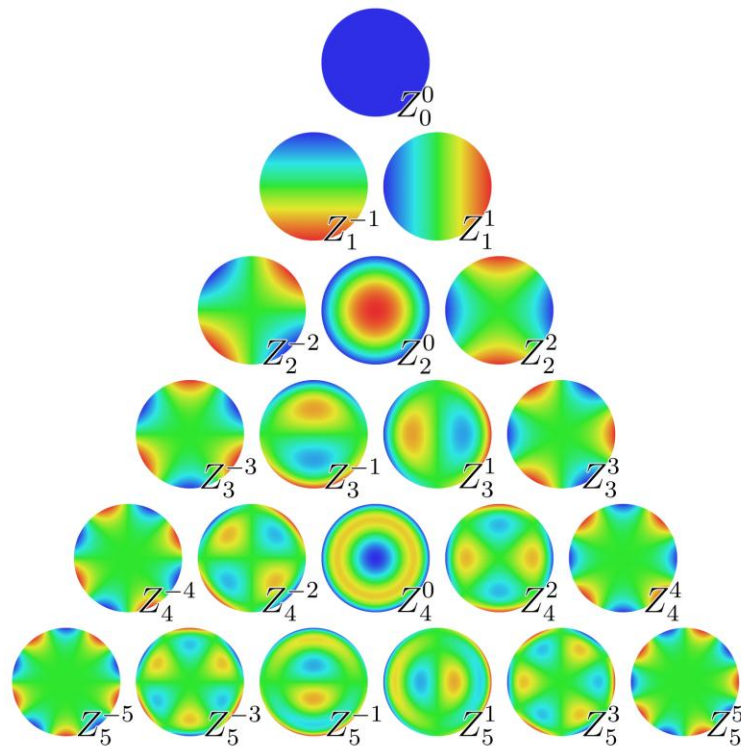


Figure 2.1.3: First order Zernike Polynomials

2.2. Atmospheric Theory

2.2.1. General Theory

The Optics research field is at the same time one of the oldest and one of the more actual fields. As a general subject, it refers to the study of the light, its properties, and its interaction with the mater. In this section we will only center in a small part of the field, introducing the basic concepts regarding the topic used during this work.

An optical communications system is one that uses signals at optical wavelengths to transmit information from one point to another. We can classify this big group in two subgroups, depending on the medium used to transmit the signal. In the first group are the wired transmissions, which go through fiber; and in the second, the wireless ones, which transmit through the atmosphere or free space. This work is centered in this second group, and a very simplified version of an ideal system of this kind could be the one represented in Figure (2.2.1).

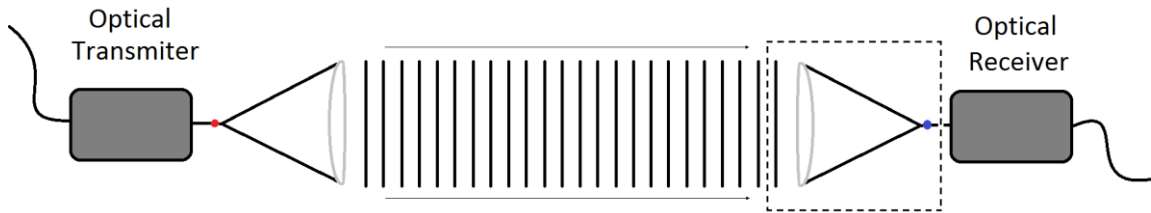


Figure 2.2.1: First approximation of a wireless optical communications system

With the image of a complex field in mind, and assuming that the distance between emitter and receiver is far enough to make the plane wave approximation, it is already possible to present a simple expression for the propagation of an Electro-Magnetic wave. (Equation 2.2.1)

$$E_x = Eme^{-\alpha z} \cos(\omega t - \beta z + \theta)$$

Equation 2.2.1: Expresion of the electric field for a plane wave propagating in direction z, with the Electric field only in the x axis. (α =Real part of propagation constant. β =Imaginary part of propagation constant. ω =Angular frequency. θ =Phase shift. η =Intrinsec Impedance).

The magnetic field would have similar characteristics but being orthogonal to both the direction of propagation and the electric field. For the moment it is not important to understand the whole formula, but to understand that the field will be a sinusoid function varying both in time and space.

Knowing that this is the behavior of the field at each point, it is possible to introduce the concept of wave-front. Although it can be a complex concept, it is already in the mind when thinking of propagating waves, imagining them as vertical lines between the transmitter and the receiver as in (Figure 2.2.1). The wave-fronts are surfaces perpendicular to the propagation line for a fixed time and space location (Figure 2.2.2).

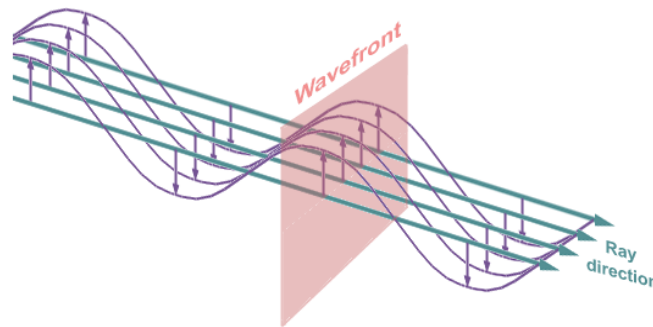


Figure 2.2.2: Wavefront concept

These wave-fronts propagate through the medium and eventually will arrive to the aperture, where they will be focused to an optical sensor or coupled to a fiber. A detailed version of the squared area of the receptor in (Figure 2.2.1) is viewed on detail in (Figure 2.2.3).

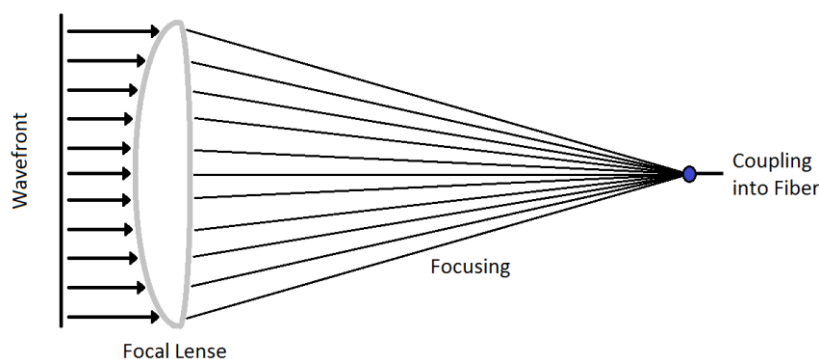


Figure 2.2.3: Field focusing

When the wave-front reaches the aperture, it can be concentrated in a smaller beam by the telescope, but in the end it is focused either to a sensor or directly for fiber coupling. The focusing concept is quite intuitive, and when the phase is not distorted and the system is well designed it is quite straight forward: In concentrates (almost) all the energy in the **aperture** in one single point, which will be the **focus**. The physical phenomena causing this can be seen more in detail in [GOO68], but the most important conclusions can be seen below.

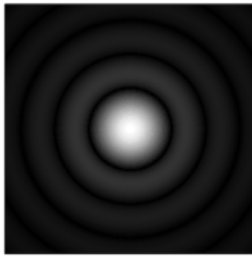
There are many types of lenses, and the effect they have in the field depends in great measure of their geometry, which determines the focal length f . The field passing through it will suffer a phase transformation, whose impact in the observed field will depend of the relation of the distance of observation with respect to f . In the case of this distance being, in fact, this focal length, the mathematical operation that is performed here can be approximated as a **Fourier Transform** of the field. This transformation is defined by the expression (Eq. 2.2.2), and has been already discussed in Chapter 2.1.1. It is a very important expression for the algorithm, as it will be seen.

$$\mathcal{F}\{g(x, y)\} = \iint_{-\infty}^{\infty} g(x, y) e^{-j2\pi(xf_x + yf_y)} dx dy$$

Eq. 2.2.2: Fourier Transform Expression

In the case of the system being **off-focus**, which means the distance from the focal plane to the aperture is different than f , the expression will be much more complex. However, in this document will be always assumed that the system is correctly focused.

For an unaberrated wave-front and a circular aperture, the result will be what is known as an Airy disk (Figure 2.2.4). Is not important to know the exact expression, but it is the fact that it concentrates most of the power in the center, like it was said previously.



$$I(\theta) = I_0 \left(\frac{2J_1(ka \sin \theta)}{ka \sin \theta} \right)^2 = I_0 \left(\frac{2J_1(x)}{x} \right)^2$$

Figure 2.2.4: Airy disk and its expression

This means that if we look at the **Intensity** (square of the amplitude) in the focus plane, we will see very high values near the center, and these values will decrease very rapidly when we go away from the central point.

2.2.2. *Aberrated Fields*

When there are zones with different temperatures, they interact, creating zones with different index of refraction known as **turbulence cells**. This index of refraction change causes the parameters α and β from (Equation 2.2.1) to change. And this change translates in the fact that the signal passing through each cell will end up with different amplitude attenuation and different phase (Figure 2.2.5). This means that in general, when looking at the wave-front in the aperture, the phase will not be plain anymore. Also, there will be amplitude variations in the aperture that are directly measurable but cannot be corrected. However, in general these amplitude variations have less impact in the focus than the phase ones.

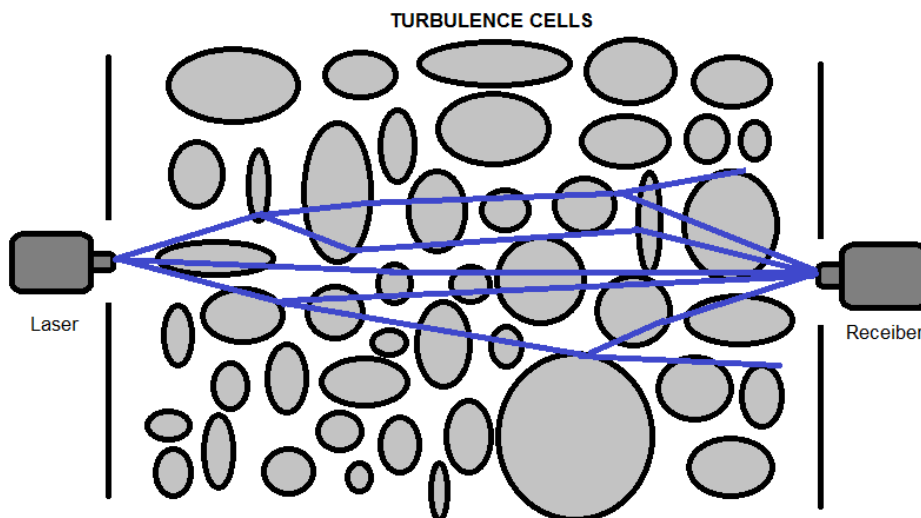


Figure 2.2.5: Transmission through atmosphere

When the wavefront is not plane, (Equation 2.2.2) must be used to determine the value of the field at the focus, because it is no longer direct. While previously there was an Airy function, now there will be a more complex expression affected by the aberrations introduced by the atmosphere. Looking at the intensity of this focus field, what can be seen is an **Speckle Pattern** (Figure 2.2.7).

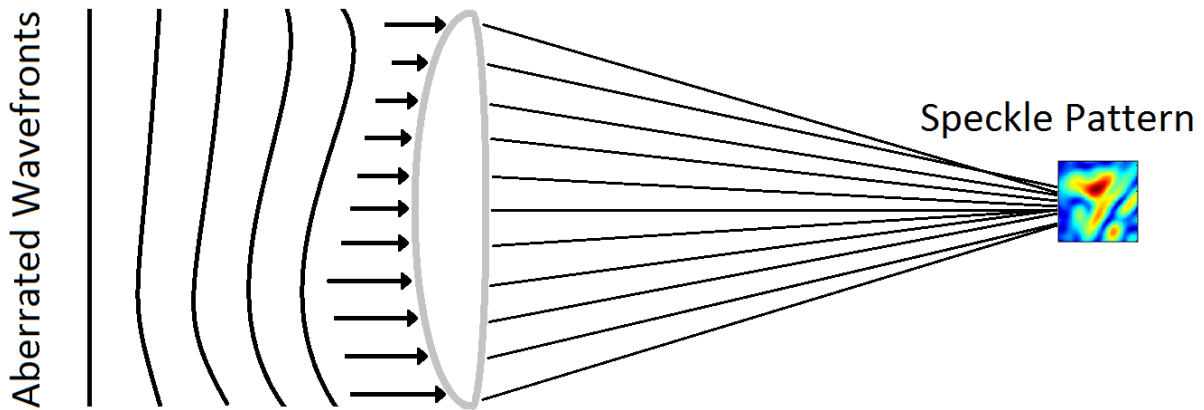


Figure 2.2.7: Aberrated field Focusing

These distortions are affected by different atmospheric factors, as the humidity and the velocity of the wind in the different layers. But there is a factor that has a huge importance especially in LEO communications: The elevation angle. This effect can be clearly seen in (Figure 2.2.8)

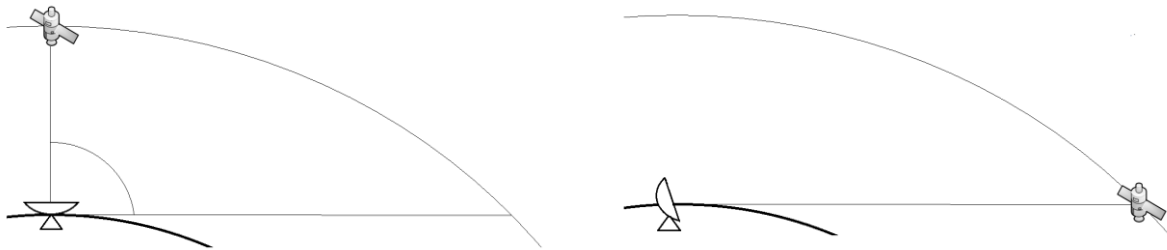


Figure 2.2.8: Earth to Satellite link with and without elevation angle of 90° (left) and 0° (right)

The lowest the elevation angle is, the more atmosphere the field must cross from the satellite to the ground-station. This implies a higher effect of the turbulence, and possibly some singularities named branchpoints, which make the detection and correction of the phase much more difficult.

Like it has been said before, a branchpoint is a discontinuity in the phase of the signal that appears when the amplitude of the signal goes to zero. If looking in the direction of propagation, these points can be seen as a helix (Figure 2.2.9). When looking at a wave-front, these discontinuities causes that tracing a random closed circuit around it a branchpoint will result in a jump of $\pm 2\pi$.

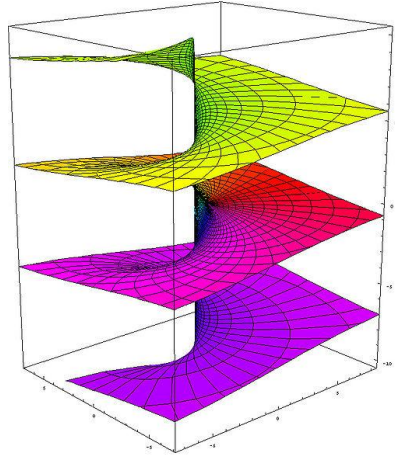


Figure 2.2.9: Branchpoint representation

In satellite communications these points tend to appear in cases where the elevation angle is very low, which implies that there is more volume of atmosphere between the satellite and the ground station, introducing higher distortions. These discontinuities difficult the correction and also the analysis of the signal, as for example making appear higher Zernike orders due to the discontinuities introduced.

Regarding the aperture size, there is another parameter that should be defined: The coherence length, or Fried parameter r_0 , and its relation with the diameter of the aperture, r_0/D . The r_0 can be defined as the limit for an area in which interior the mean differences in the phase between its points are inferior to 1 radian, and can be calculated many ways. The most general is the presented in (Equation 2.2.3)

$$r_0 = 0.185\lambda^{6/5} \cos^{3/5} z \left[\int (C_n^2 dh) \right]^{-3/5}$$

Equation 2.2.3: Fried parameter in function of the wavelength λ , the zenit angle z (90°-elevation angle), and C_n the refractive index structure constant

Like it can be seen, it is a factor that depends of the atmosphere that is crossing, in this case by having to integrate C_n , unknown variable which depends on the structure of the whole atmosphere. Therefore, making this calculation is not feasible in a satellite communications environment. Because of this, some approximations are made [G11b] for being able to estimate this parameter with the data of a single field intensity speckle pattern (Equation 2.2.4).

$$r_{0-st} \approx \frac{8}{\frac{1}{D} + \rho_{FIS-tr-st} * \frac{10}{0.59 * \lambda * f}}; \quad r_{0-st} \leq D$$

Equation 2.2.4: Short-term fried parameter in function of the wavelenght λ , apperture diameter D , the focal length f , and the ρ extracted from the Gaussian that best fits the Focal Intensity Speckle pattern

Inside an aperture with r_0/D greater than one, the phase can be considered mostly constant or with low distortions. However, with an aperture greater than that, greater distortions should be expected (Figure 2.2.10). This is a very important parameter to calculate, as it tells us how distorted we can expect the phase to be. This could induce to think that the smaller the aperture the better. But the smaller the aperture, the less power will be inside of it, so a compromise should be made in this aspect.

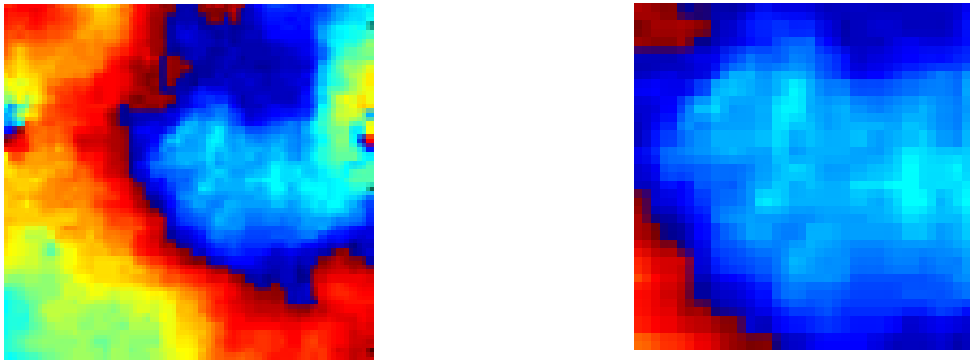


Figure 2.2.10: Aperture phases for different diameter sizes

To determine the quality of a field, several parameters can be defined. In this document, the ones that will be used will be the Strehl ratio (S) (Figure 2.2.11) and the Heterodyne Efficiency (H). The Strehl factor is defined as the quotient between the maximum Intensity in the focus of the field, and the maximum of an unaberrated signal with the same total power. The Heterodyne Efficiency, on the other hand, represents the matching of the whole field with respect to this reference.

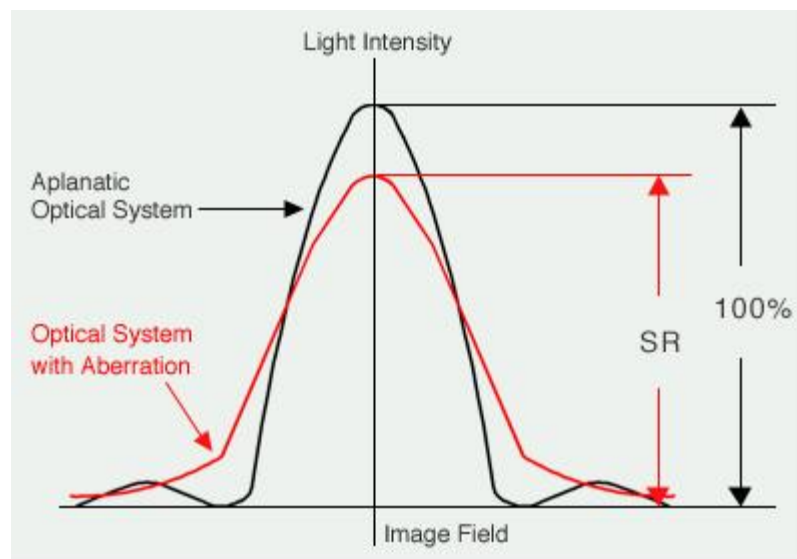


Figure 2.2.11: Concept representation of the Strehl ratio

The main difference between the two parameters is that the second one gives information of how centered the signal is, and how similar the shape is to the ideal case. On the other hand, the first one only reflects how concentrated the power is, independently of the shape and the position of this power. However, centering the signal is a problem which normally can be solved without many difficulties, so in many cases the information of the **Strehl ratio** will be more representative than the Heterodyne Efficiency.

2.2.3. Adaptive Optics

The correction of the signal passes by knowing the shape of the aperture field. If it is known, the wave-front can be corrected to get a plane one. That way, the focus spot will become the desired Airy pattern, and the coupling would be optimum. The problem is that this information is in the phase, and the sensors are only capable to detect the Intensity, and from this the Amplitude. And this happens both in the aperture and in the focus.

One of the most extended ways of solving this is using a **Wave-front Sensor**, which lets the complete scheme of the system as shown in Figure 2.2.12. This scheme may serve to illustrate the behavior of an optical receiver.

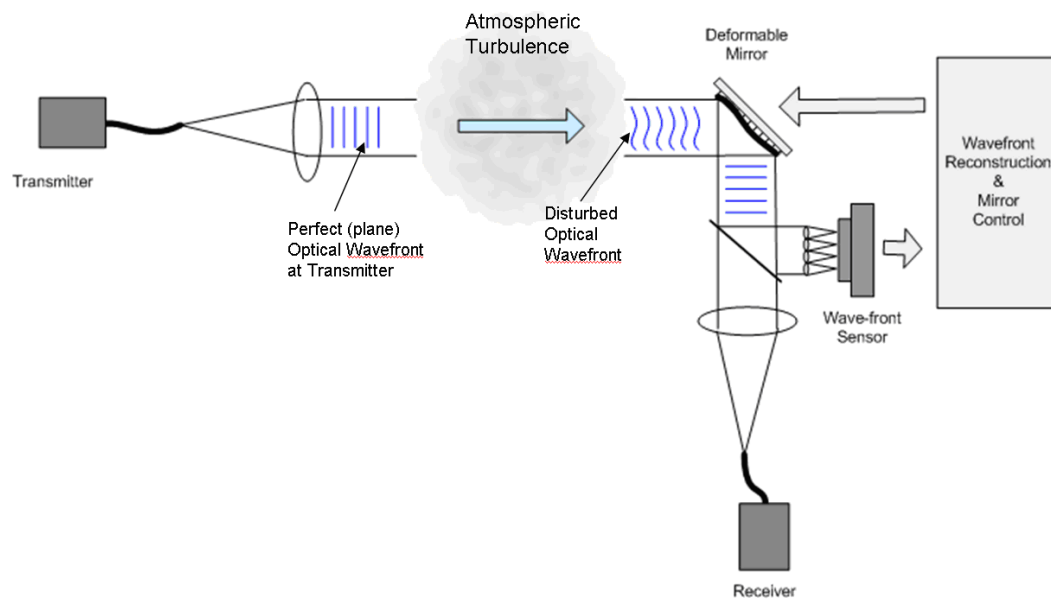


Figure 2.2.12: Scheme of an Adaptive Optics system with wavefront sensor

The first step is not in the Scheme, and it may not be considered part of the AO system, but it is important for the algorithm: The telescope must be correctly aligned. The distance is of around 500km, so any error pointing the telescope may result in a complete loss of the signal. Beacon beams are generally used and detected and pointed by secondary systems. But nevertheless, some fine adjustments must be made to have a maximum efficiency, especially when the telescope is already aligned and must simply follow the path of the satellite through the sky. This is made with a tracking camera. This camera is placed after a focalization of the signal, as the receiver above, so the pattern observed in the camera will be equal as the one at the fiber coupling. The APHEFIS algorithm, as will be seen, extracts information from this camera, so it is important to note that even if it is not in the diagrams of AO systems, it is generally present anyway.

The first step in the loop is the wave-front sensor. The incoming wave is divided in two by a mirror, a part of it is focalized, and a part of it reaches the sensor. There are different types of sensors, but one of the most used is the Shack-Hartman Sensor (Figure 2.2.13).

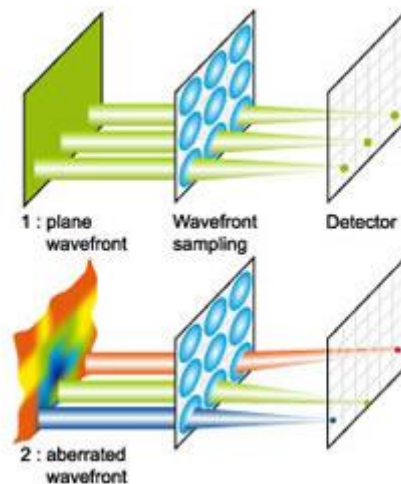


Figure 2.2.13: Shack-Hartmann Sensor

The data measured by the sensor is passed to a computer, which makes the necessary calculations. Measuring the displacement of each of the spots created by a matrix of lenses, the gradient of the wave-front at this point can be calculated. Also, it is known that the difference of phase between two points is proportional to the difference of phases between them, with the proportionality constant depending only of the geometry of the physical distribution. Using this information, we can estimate the phase at each sub-aperture, information that can be used to correct the aperture field.

Once the phase has been reconstructed, it is transformed in orders to the deformable mirror to correct the signal. This deformable mirror is a mirror with an adaptable surface, generally composed by many other small mobile mirrors. With this, each part of the incoming beam can be reflected differently, compensating the difference of phase in the wave-front.

After that, the loop goes back to the start. The atmosphere changes, and the distortions with it, so the sensor measures phase differences, the computer processes and gives orders to the mirror, and it shifts its form to correct the new wave-front.

However, the Shack-Hartmann sensor is inherited from Astronomy, and for that it has the limitations already mentioned. It was thought for working in fixed ground stations placed in very specific locations: High altitudes, low moisture, and high

percentage of clear sky (no clouds). Moreover, it was thought for looking at objects near the zenith, where less volume of atmosphere must be crossed. All of this makes the sensor very good for low distortions, but not reliable in the presence of the high order aberrations and branchpoints common in low elevation angles. Because of those weaknesses of the astronomic inherited systems when applied in communication environments, new technics must be developed.

More information about the SHS can be found, for example, at [HAR98], and a more complete analysis and comparison between the different techniques used nowadays is made at [KNA10], among others.

2.3. Other tools

2.3.1. Segmentation Tools: Watershed Algorithm

One of the most important parts of the project deals with the need of finding the zones which have similar characteristics and so are prone to have similar phases. The determination of these zones is actually made by the watershed algorithm, which will be described here. Much literature about the topic has been written and can be consulted for a more exhaustive explanation, like [GON08].

For having a visual image of what the watershed algorithm does, we have to look at our intensity image as a 3D image (Figure 2.3.1).



Figure 2.3.1 Left: Original grey scale image, Right: Topographic vision of it (Images from <http://cmm.ensmp.fr/~beucher/wtshed.html>)

If we look at it as a topographic map, it is easy to imagine the peaks here as mountains, with valleys between them. The objective in this case, is to isolate in

different zones the peaks, or what is the same, the mountains. However, the watershed algorithm separates in terms of the minimums. Therefore, the image must be inverted, making the mountains become the valleys. In this case it is not as similar as a topographic map as before, but the comparison still serves to give an idea of the behavior of the algorithm.

Now, there are different kinds of watershed algorithms and even the general behavior is the same, the concept can be very different. For starters, the concept of the most general case will be explained, and afterwards the variations will be commented.

Imagine that a hole is made in each of the minimums of the image, and water is pumped from there. All the holes are pumped at the same time, so the level of one minimum will not start going up until the general level reaches its height. The water will continue rising until, eventually, at some points the water of two valleys would merge if the level is increased further. Here, a dam is built to prevent the merging, and the process continues until the only thing observable from above are the dams or the water. At this point, the algorithm is done: The points catalogued as “dams” will be the frontier points or “watershed lines”, and the water will have filled individually every watershed zone.

In this kind of approach, another easily visual image can give us the idea of which zone a certain point belongs, looking at the hypothetical behavior of a drop of water (or a ball) let in this point. If the water would fall undoubtedly to a minimum, it belongs to the zone of that minimum; while if it has the possibility to fall to two or more minimums, belongs to the watershed lines (Figure 2.3.2).

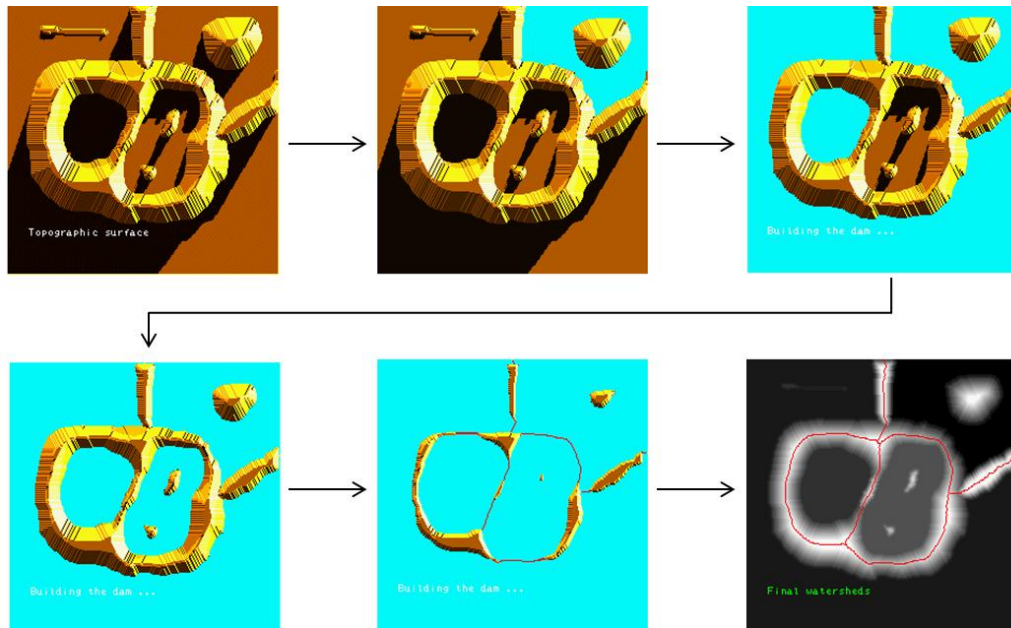


Image 2.3.2: Example of watershed algorithm behavior

The previous approach works fine for most segmentation applications, but has several lacks for some applications, like the possibility of making an over-segmentation. In an image with lot of noise, there will probably many minimums that don't belong to the background image we want to segment. Or maybe they are present, but we only want to have the zones related to the lowest minimums, or the biggest zones. In this case, we should use a variant of the watershed algorithm which works with what is known as markers (Figure 2.3.3). For the watershed algorithm, we can interpret the markers as the holes from where the water will be pumped. If we don't make a hole in one of the valleys, the water level will not start rising in it at the beginning, but eventually it will be poured from above coming from other peak. This way, the zones without a marker will be absorbed by their neighbors, and so problems like the over-segmentation can be prevented.

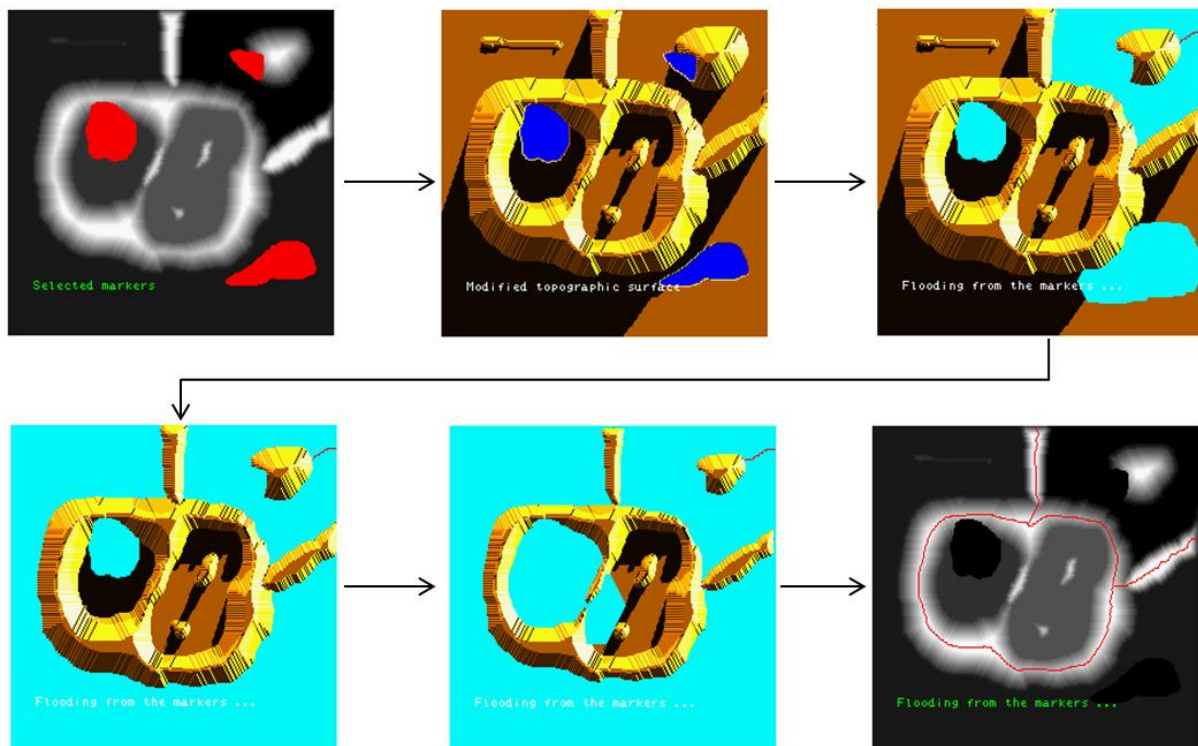


Image 2.3.2: Example of watershed algorithm with markers

2.3.2. *Artificial Neural Networks*

In the machine learning algorithms, the neural networks are like the optics in physics. Their origins go back to 1943, with the computers still in embryonic state, when McCulloch and Pitts developed the first neuronal model. Unluckily, being in such a primitive stage, this research could not advance all that could have, and it was not until the late 90s when it got a solid form.

The general idea, which gives the algorithm its name, is to create a computational system that emulates the behavior of the neurons of the brain. In it, each neuron has connection to one or more neurons, which have connection to one or other neurons, etc. When they must process information, the “result” of each neuron goes to the next ones, then to the next ones, and so on until the “final result” is obtained. Lots of versions of Artificial Neural Network (ANN) have been developed in the last years, which can differ in almost every aspect but the general idea. For a more detailed analysis, specialized bibliography like [GUR97] can be consulted.

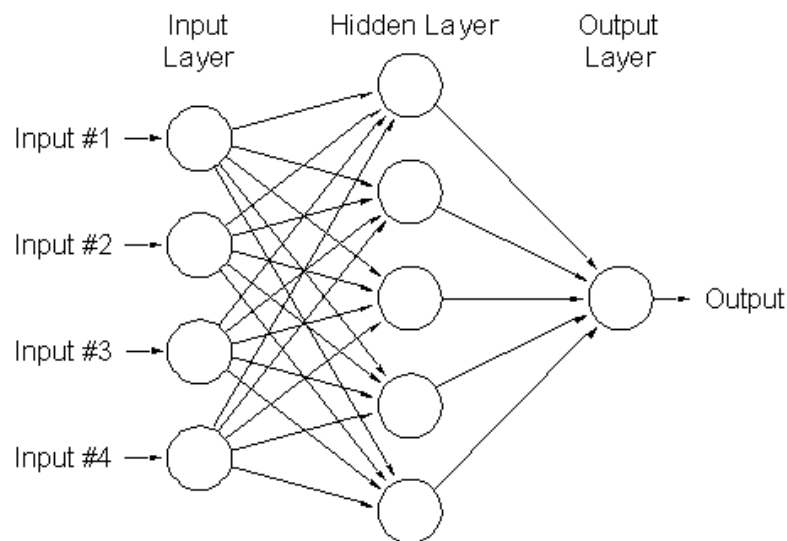


Figure 2.3.3: General diagram of an ANN

In an ANN the behavior can be synthesized with the previous diagram (Figure 2.3.3). As a notation, the Input layer are the values we previously know, the output layer is what the system tells us after analyzing the data, and the hidden layers (here is one, but can be any number we want) are a black box whose values we don't know.

First we have our input layer of nodes, which is simply the inputs we have. The information of this layer is used as an input by each node of the first hidden layer, which in function of them outputs a value. The output of all the nodes of the first hidden layer goes to the next layer, in which the same procedure is repeated, and so on until the output layer, which are the signals we want to obtain.

Almost all ANN actually in use follow this general concept. In what they differ, is in the process each node makes. Again, as a general concept we can say that they follow the scheme showed in (Figure 2.3.4).

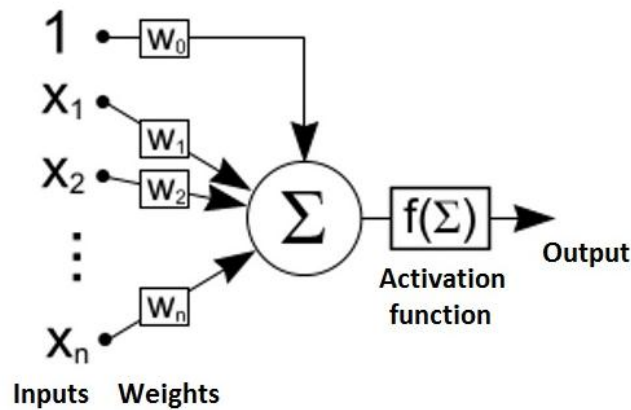


Figure 2.3.4: Behavior of a Node in an ANN

Each of the inputs, plus a constant, are multiplied by a different weight and added, obtaining a number which will determine the output of the neuron after an activation function is applied to it. This activation function can be a user defined function, but generally only three options are used for it, because they have a good behavior and also because they have a known way to make the back-propagation to determine the coefficients, which will be discussed later in this section. This options are a linear function, a step, and a sigmoid or logistic curve. And from this three, the most used for its general good behavior is the third, which equation and its derivative corresponds to the expression in (2.3.1) and (2.3.2), respectively.

$$P(t) = \frac{1}{1+e^{-t}} \quad (\text{Eq 2.3.1})$$

$$P'(t) = P(t)(1 - P(t)) \quad (\text{Eq 2.3.2})$$

This function can be used to represent many characteristics in other ambits, like in population growth models, for its shape tending to zero for t tending to $-\infty$ ($t < 0$), and to 1 for t tending to ∞ ($t > 0$). This expression can fit quite acceptably a variable that starts growing after a threshold is surpassed, has a linear zone around zero, and arrives to saturation after a second threshold. In ANN, the saturation parts are the most used: The output should be very close to one or to zero for most inputs, although intermediate values are accepted for not-so-clear inputs.

With this information, a full neural network can be obtained if the relationship between the parts is known. However, the strong point in an ANN resides in its capability to extract these relations. This is the role of the algorithm known as back-propagation algorithm, which is able to get the different weights that at this point are the only unknown part of the system.

The process is quite complex, and there are lots of documents regarding this topic. A more detailed explanation of it, like any other aspect of ANN, in addition to the previously commented bibliography, [ROJ96] can be consulted. The idea behind it is the following: First, some random values are assigned to each weight. With them, an iteration with all the known entries is done, and the error between the actual outputs and the desired ones is obtained. The objective here is to minimize this error, so for each node the derivatives with respect to each entry are estimated, and are used to update each of the weights. This process is propagated back (from here comes the name) from layer to layer until the initial layer is reached, and a new iteration is started then.

3. Phase Estimation from Focal Intensity Speckles algorithm

3.1. Introduction

The basic idea behind the APHEFIS algorithm is that it is possible to obtain an estimation of the (not observable) phase in the focus from its (easy measurable) intensity, using the fact that there is a correlation between those characteristics in that plane that is not present in the Aperture one. Once the phase in the focal plane is estimated, it can be used to obtain the field on the aperture through the Inverse Fourier Transform (IFT), like seen in Chapter 2. This field phase is then used to correct the wave-front through a deformable mirror. The basic scheme can be seen in (Figure 3.1.1)

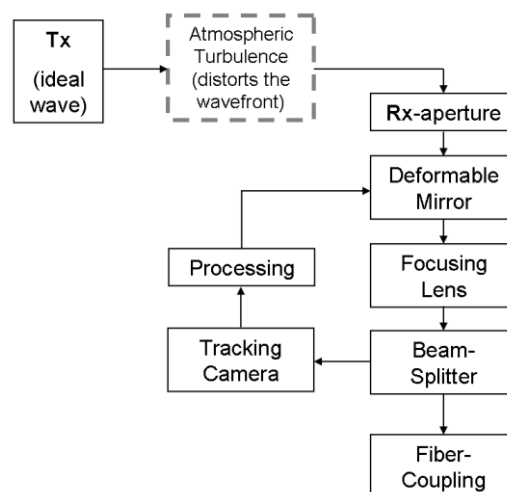


Figure 3.1.1: Basic scheme of the described method

The focus intensity measurement is made by a camera, which images the focus speckle pattern (FSP) of the incoming beam. A focus camera is in many cases also used for tracking purposes, helping to center the beam for a correct coupling. This means that the system does not add any hardware complexity to the receptor system. And in fact reduces this hardware complexity, as it eliminates the need of

another complex tool to measure the hardware distortions, like the classical Shack-Hartmann Sensor.

The algorithm is based in four assumptions:

- 1) There is a **correlation between the intensity and the phase in the focal plane** that can be determined.
- 2) A **two level approximation of the phase in the focus** is enough to make a good estimation of the aperture phase.
- 3) The phase tends to have the **same sign phase around the intensity peaks**.
- 4) The phase of the **higher intensity peaks holds more information** than the lower ones.

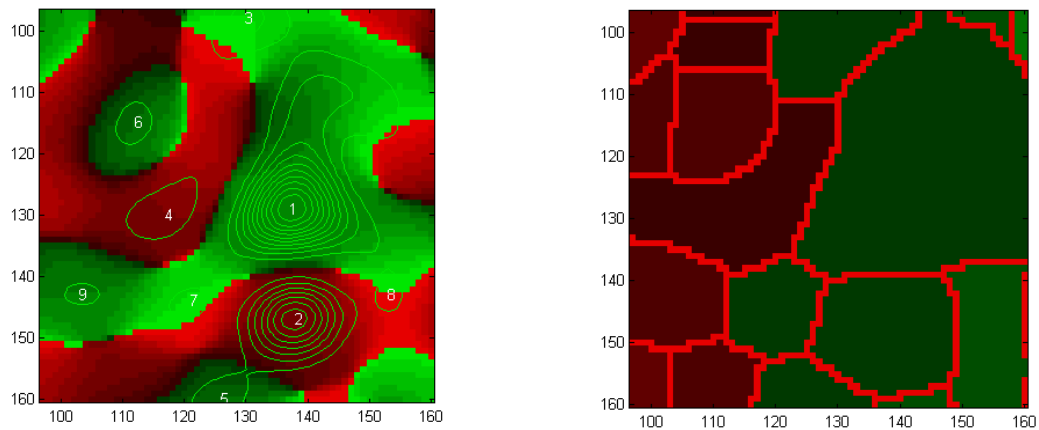


Figure 3.1.2, Left: Focal intensity distribution (light green contours) and phase distribution (color scale: light to dark green for $+\pi$ to 0° , dark to light red for 0° to $-\pi$)
 Figure 3.1.2, right: Watershed zones of the intensity

The basic ideas of the algorithm can be easily visualized in (Figure 3.1..2). The green contours in the left image represent the focus speckle pattern and the color represents the phase, with green for positive and red for negative. The first thing that can be observed is that under a low level of distortions, there is a clear correlation between the intensity levels and the phase (**Assumption 1**), as the contour lines follow approximately the same pattern as colors. However, this doesn't seem enough to predict the whole phase, as this correlation is looking at the colors (which represent the sign) and not at the whole phase. So it is considered that a two levels evaluation of the phase should be sufficient (**Assumption 2**) to obtain a good approximation of the phase through the back-transformation. However, processing the phase of each point separately would be both computationally inefficient and

provably take to the limit the assumption 1. So a segmentation tool, in this case the watershed algorithm, is used to determine zones which will have a constant phase (**Assumption 3**). This can be seen in the second image, where the red lines separate the different zones. And last, due to computational reasons and to the fact that the low power zones will have not so clear characteristics, so processing them could have a negative impact in the overall efficiency of our system, it's assumed that only high power zones should be treated (**Assumption 4**). This is not seen in the pictures as in they only this high power zones are seen, but it will be seen in the next sections. In the following pages the environment for the tests will be explained, and later each assumption will be tested and the obtained results presented.

3.2. Test Environment

To test the behavior of the system, a software which works as a test environment has been developed. The software has three main parts: The GUI, the AO algorithm, and the connecting part (also known as bridge), which simulates the outside world that the algorithm sees.

The GUI is the main interface with the user. Although the two other big programs can run independently, the huge quantity of variables they use and the size of the table they return make it not very advisable.

This graphic interface has two main parts: A table to show the results returned by the main program and a window where most of the variables can be fixed, and where the different modes of work can be chosen. These modes are the simulation mode, the ANN training mode, and the ANN test mode.

The simulation mode is the most general one, in which the main program runs analyzing the behavior of the system with the selected fields, and storing in memory the data it returns. Depending on the selected options, the simulation can run to get atmospheric and system data for a posterior ANN training, or to use the actual parameter for making a real AO system correction.

The ANN training mode starts a second program, whose work is to optimize an ANN to fit the data currently stored in memory. Depending of the options selected, it can further optimize a previously trained system, or create a new one starting from random weights.

Finally, the ANN test mode is used to test a Neural Network with the data stored in memory. It must be noted that this test only evaluates the sign assignation, returning the percentages of signs correctly determined, but it does not make a complete simulation and therefore will not change the table of results or the stored data. This option is only used to rapidly find if an ANN trained with certain data is acceptable to evaluate data with another characteristics.

A detailed guide for using the GUI is given in APENDIX C, and a picture of the main window and presentation table is shown in (Figure 4.3)

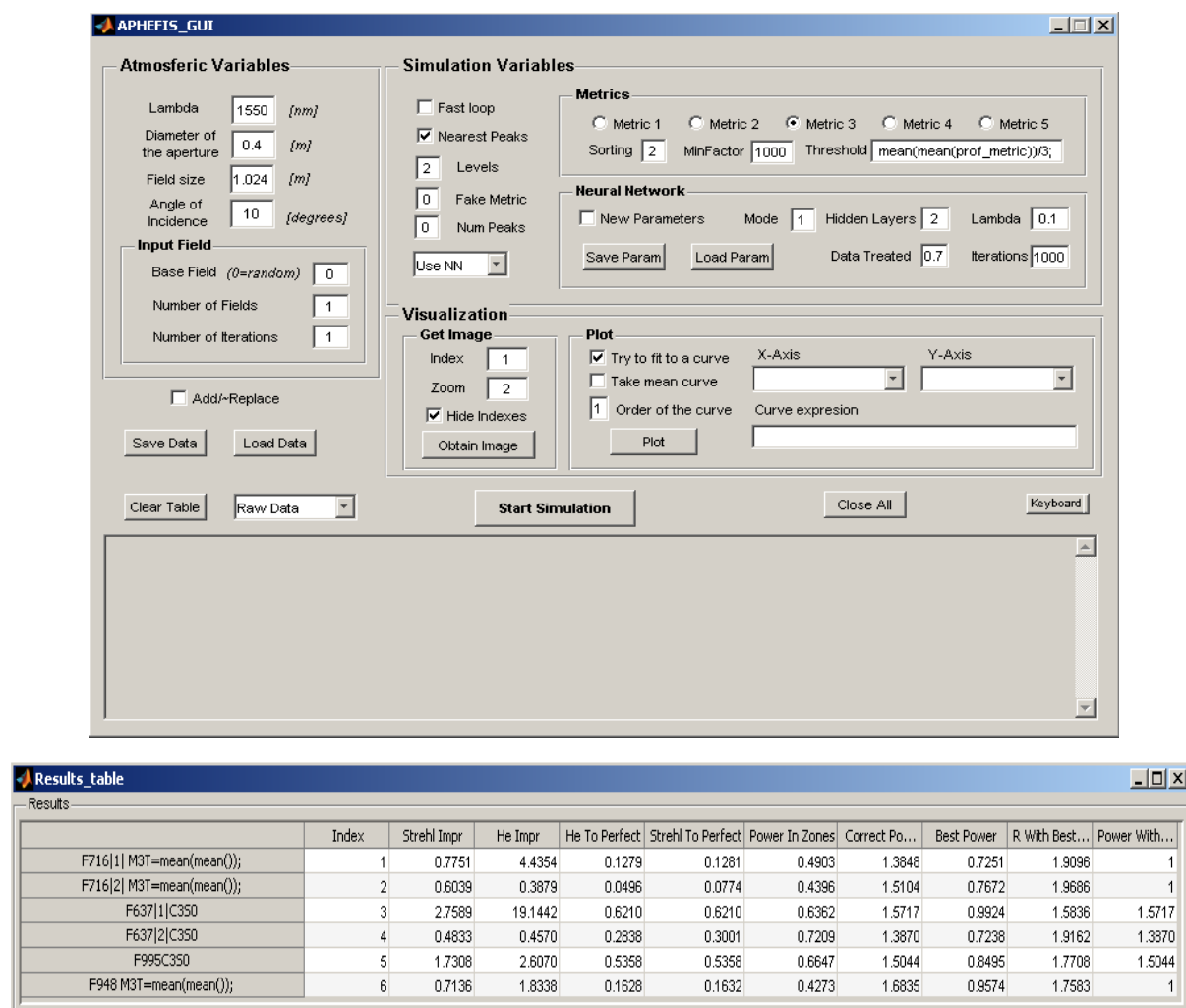


Figure 3.2.1: GUI main window (up) and presentation table (down)

The second big part of the program is the called “bridge”, as its role is to connect the data with the phase determination algorithm, making this second one work as if it was in a real environment. This second part is the one that more files include (a list of all the files and its functions can be found in APENDIX B), but the main one is the called APHEFIS_bridge.m. The input parameters of this file are directly given by the GUI, and with the data created with PiLab.

PiLab is a DLR toolbox whose function is to simulate atmospheric distortions based on different input characteristics, like the incidence angle, and obtain the resulting fields. For this work, a total of 4000 PiLab simulations have been used, a thousand of each of the incidence angles 10° , 20° , 30° , and 90° .

With the data given by the GUI, the bridge starts by opening the selected PiLab fields, cutting them to fit the selected aperture size and masking them to fit a circular lense. After that, an analysis of the intrinsic field characteristics is done. This analysis includes, among others, the determination of the existence of discontinuities (**branchpoints**) and its numbers, and the **estimation of r_0** (estimation because its accurate determination is not possible for a single isolated image). If it was selected, here also the **Zernike coefficients of the input** are calculated. These values are vital for making a correct posterior analysis.

After that, the **FFT** is applied to simulate the focusing effect of the lens, and new characteristics are extracted. In this case, the most important ones are the quality variables, as the **input SR** and HE. After that is done, the FSP is given as an input to the true **algorithm program**, named APHEFIS_correction.m.

The concrete behavior of the program falls into the implementation topic, so will not be discussed here. However, the general behavior is general to all of the implementations. First, the watershed algorithm is applied to the FIS. After that, the highest peak has its sign assigned to either value, and the characteristics of each zone are extracted. After that, each zone is evaluated and its phase sign determined by the neighboring ones and how they are related. When comparing two zones the basic decision that must be made is if the zones should have the same sign or the contrary. This behavior continues until every zone has its sign assigned.

How this is done will be discussed in other chapter, but what must be clear is that this program, as mentioned before, gets as an input the FSP and gives as an output the corrections to be done by the deformable mirror. At the moment it returns only the focal phase, and the IFFT must be calculated to obtain the corrections, but this is only for clearness on the program writing and can be easily solved for a definitive version.

When the phase estimation in the aperture is done, the correction is applied by directly subtracting the estimated phase to the original one. With this, only the prediction errors remain. And as the environment here is static, so the input doesn't change while the correction is being calculated, this error is the phase of the input aperture field after correction. If it was asked to, the **Zernike coefficients of the output** are calculated too. And either way, the field is focused again to obtain the **Output SR** and HE.

Once all the parameters are calculated, they are returned to the GUI in the form of a table. There some other statistical parameters are calculated in function of the values returned, and the data is presented in the window designed for that purpose.

3.3. Considerations regarding the symmetries of the phase

For unitary amplitude signals, many symmetry properties can be extracted. In this case, a signal $f(x, y)$ can be expressed as $f(x, y) = \cos(\varphi(x, y)) + j \sin(\varphi(x, y))$, and $\varphi(x, y)$ can also be decomposed in its components $\varphi_o(x, y)$ and $\varphi_e(x, y)$. Knowing that the phase is restricted to the interval $\pi > \varphi(x, y) > -\pi$, some properties of this trigonometric functions can be used here also.

$$\cos(a + b) = \cos a * \cos b - \sin a \sin b \quad (\text{Eq 3.3.1})$$

$$\sin(a + b) = \sin a * \cos b + \sin b \cos a \quad (\text{Eq 3.3.2})$$

Some general function like that $\cos(\varphi_e)$ is even, and $\cos(\varphi_o)$ is also even, while $\sin(\varphi_e)$ is even, and $\sin(\varphi_o)$ is odd can be very usefull as well as some trigonometry functions like (Eq. 3.3.1) and (Eq. 3.3.2). For simplicity and compactness, I will eliminate the dependences in the expressions ($\varphi_o(x, y)$ will be expressed as φ_o), but it should not be forgotten that each letter in this case will be a function, not a variable or a constant. So, the expression:

$$f(x, y) = \cos(\varphi_e + \varphi_o) + j \sin(\varphi_e + \varphi_o) \quad (\text{Eq 3.3.3})$$

Will be expressed using (Eq. 3.3.1) and (Eq. 3.3.2) as:

$$f = \cos(\varphi_e) \cos(\varphi_o) - \sin(\varphi_e) \sin(\varphi_o) + j(\sin(\varphi_e) \cos(\varphi_o) + \sin(\varphi_o) \cos(\varphi_e))$$

$$(\text{Eq 3.3.4})$$

Now, applying that *even * even = even*, *odd * odd = even*, and *even * odd = even*, this can be further simplified to get the expressions of the even and odd parts separately.

$$\begin{cases} f_o = \sin \varphi_o (j \cos \varphi_e - \sin \varphi_e) \\ f_e = \cos \varphi_o (\cos \varphi_e + j \sin \varphi_e) \end{cases} \quad (\text{Eq 3.3.5})$$

With these expressions many conclusions can be extracted, but having the product of the different components it's difficult to see them. So, we can use another trigonometric property easily extracted from 5.1.

$$\cos(a) = 2 \cos^2 \frac{a}{2} - 1 = -2 \sin^2 \frac{a}{2} + 1 \quad (\text{Eq 3.3.6})$$

And applying it to (Eq. 3.3.5)

$$\begin{cases} f_o = j \sin \varphi_o - \sin \varphi_o \left(2j \sin^2 \frac{\varphi_e}{2} + \sin \varphi_e \right) \\ f_e = \cos \varphi_o - \cos \varphi_o \left(2 \sin^2 \frac{\varphi_e}{2} - j \sin \varphi_e \right) \end{cases} \quad (\text{Eq 3.3.7})$$

Applying it to (3.7) and decomposing it in 2 parts for making it easier to operate with it as $\mathcal{F} = \mathcal{F}_e - j\mathcal{F}_o$, and with the substitution $\theta^f = 2\pi(xf_x + yf_y)$ for compactness reasons:

$$\begin{cases} \mathcal{F}_o = \iint_{-\infty}^{\infty} \left(j \sin \varphi_o - \sin \varphi_o \left(2j \sin^2 \frac{\varphi_e}{2} + \sin \varphi_e \right) \right) \sin(\theta^f) dx dy \\ \mathcal{F}_e = \iint_{-\infty}^{\infty} \left(\cos \varphi_o + j \cos \varphi_o \left(2j \sin^2 \frac{\varphi_e}{2} + \sin \varphi_e \right) \right) \cos(\theta^f) dx dy \end{cases} \quad (\text{Eq 3.3.8})$$

With this, some very interesting conclusions can be extracted. For starters, the extreme cases must be considered.

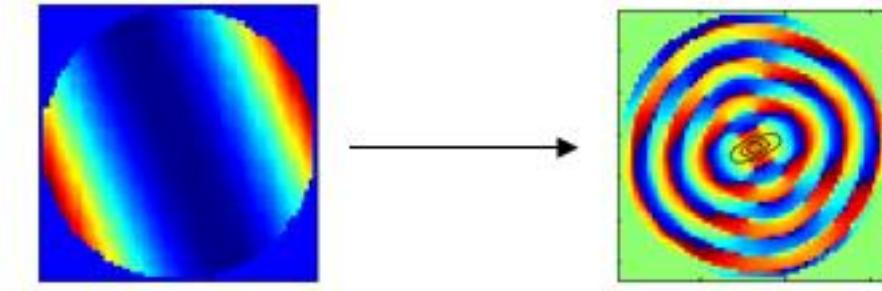


Figure 3.3.1: Transform of a pure even function

In case of the phase of the signal being purely even ($\varphi_o = 0$) (Figure 3.3.1), the transform will end up like this:

$$\mathcal{F} = \iint_{-\infty}^{\infty} (\cos \varphi_e + j \sin \varphi_e) \sin(\theta^f) dx dy = \iint_{-\infty}^{\infty} e^{j\varphi_e} \sin(\theta^f) dx dy \quad (\text{Eq 3.3.9})$$

Whose result will be an even function with arbitrary phase, as seen in (Figure 3.3.1)

For an odd function, though, the result would be:

$$\mathcal{F} = \iint_{-\infty}^{\infty} (\sin \varphi_o \sin(\theta^f) + \cos \varphi_o \cos(\theta^f)) dx dy \quad (\text{Eq 3.3.10})$$

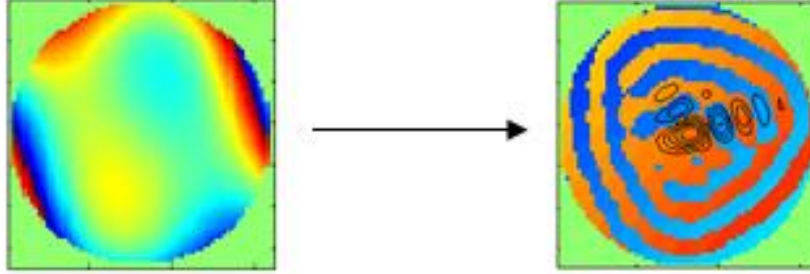


Figure 3.3.2: Transform of a pure odd function

And this is a very important expression. From this expression can be extracted that the focus field of an aperture with only odd phase components would be real (Figure 6.2). This has two main implications regarding the system.

For starters, the phase will be already leveled, because the only differences in the phase will come from the sign of the real part. This means that although normally the APHEFIS algorithm has only the capacity to make leveled approximations of the phase, in this case it will be possible to perfectly approximate the phase at the focus, making possible the total correction of the aperture field.

And, as a second conclusion, a real function will always be perfectly adapted to a watershed algorithm in absence of noise. Or, saying it more generally, the phase in the focus is totally correlated with the amplitude in it. All the changes of phase will be situated in minimums, because the intensity will always be zero when a change from positive to negative is present.

This means that for an odd phase in the aperture and relatively low amplitude distortions, the APHEFIS algorithm will be able to do a perfect correction in all of the fields. And, of course, the nearest the distortions are to this, the best the system will behave. It is because of this that the developed expressions (3.3.9) are useful, because they let us express the focus field as a sum of the coefficients coming from the odd part and the rest of them. If the odd parts are extracted, the rest could be expressed as:

$$\begin{cases} \mathcal{F}_o^r = \iint_{-\infty}^{\infty} -\sin \varphi_o \left(2j \sin^2 \frac{\varphi_e}{2} + \sin \varphi_e \right) \sin(\theta^f) dx dy \\ \mathcal{F}_e^r = \iint_{-\infty}^{\infty} j \cos \varphi_o \left(2j \sin^2 \frac{\varphi_e}{2} + \sin \varphi_e \right) \cos(\theta^f) dx dy \end{cases} \quad (\text{Eq 3.3.11})$$

And getting it all together again,

$$\mathcal{F}^r = \iint_{-\infty}^{\infty} \left(j \sin \varphi_e - 2 \sin^2 \frac{\varphi_e}{2} \right) (\sin \varphi_o \sin(\theta^f) + \cos \varphi_o \cos(\theta^f)) dx dy \quad (\text{Eq 3.3.12})$$

What is interesting about this expression is fact that it reflects that the power of the interferences will be also proportional to the part that the algorithm can correct.

However, it must be always considered that the phase is a very particular kind of data with some characteristics that make its treatment to be different of the rest. In any not cyclic signal, the only possibility for a signal to have an odd and at the same time even symmetry is to be continuously 0, because obviously 0 is equal to -0. This makes it impossible for an even signal to approximate an odd one, and for an odd to approximate an even one. However, in the case of the phases a new possibility appears: π . π is equal to $-\pi$, and of course is equal to π . With this, there are already two levels with are at the same time even and odd, and make it possible to approximate an even signal by an odd and the other way around. However, it is important to see that even if this approximation is made, in the end it is introducing high frequencies that may end up complicating the analysis (Figure 3.3.3).

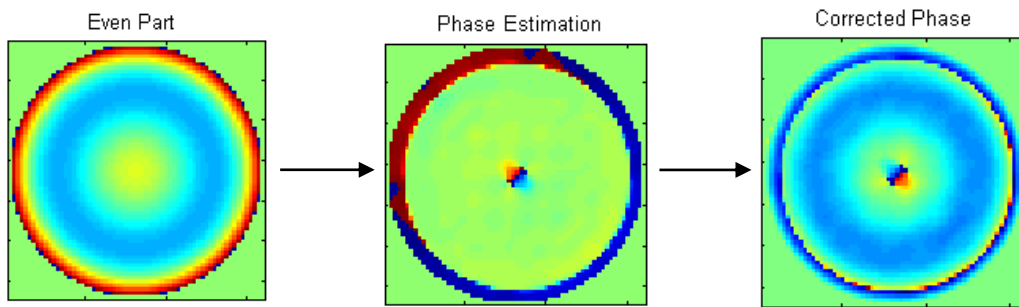


Figure 3.3.3: Aproximation of an even phase by an odd one, and result of the correction

This has another implication, which is the fact of the division in even-odd components not being enough to approximate the behavior of the system, as it can also approximate somehow even components, even if it is very roughly. Some further analysis is required.

3.4. Algorithm assumptions

In this section each of the assumptions will be tested, checking what implies making it and what are the effects in the aperture phase prediction.

For all the evaluations that doesn't say otherwise, the analysis is made by applying the assumption considerations (or the implementation) to the whole pack of 4000 fields for apertures of 0.4m, 0.6m, 0.8m, and 1m. Regarding the graphics, "*Phase Estimation*" will be the phase obtained from back-transforming the estimated focal field, and "*Corrected Phase*" will be the result of subtracting the previous to the original phase.

Regarding this analysis, some particular effects are easier to show with simplified phases. In these cases, a single (or a combination of few) Zernike polynomial with unitary amplitude will be used for graphic representation, and will be called "Simplified phases".

3.4.1. *Two levels evaluation of the phase*

One of the most important assumptions that the APHEFIS algorithm make is that a two-level approximation is good enough to represent the phase in the focal plane. In this section, this assumption will be evaluated and the behavior of a system with a perfect 2-level prediction of the phase will be tested.

For the evaluation of the two levels phase assumption, the estimation of the phase is made by leveling the focal plane one. This is made by observing the wrapped phase (between $-\pi$ and π), and approximating the phase of each point by looking at its sign, so positive phases are assigned to $\pi/2$, and negatives to $-\pi/2$. This operation is represented in (Figure 3.4.1). The black lines represent the levels of the amplitude in the focus in the right figure, and the square (the intensity) in the left.

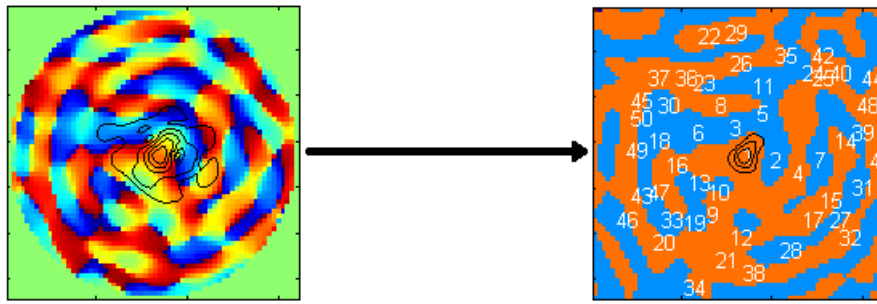


Figure 3.4.1: Two level approximation of a phase, with the highest peaks numbered

As was seen in section 3.3, this approximation has a physical meaning related with the symmetries of the aperture phase. When back-transforming the focal estimation, in this case with the 2-leveled phase, and using it to correct the signal by subtracting it, the results obtained with simplified versions of the phase are shown in Figure (3.4.2).

What must also be noted is the fact that leveling the phase always leads to discontinuities. This discontinuities lead to high frequencies. And, if not filtered, these discontinuities will make appear aliasing in the system, a factor that can degrade very much the quality of the prediction. If it is assumed that this high frequencies will have low power, it is possible, although still not recommended, to make not further treatment of the signal. However, if this is not the case, a filtering of the signal becomes necessary.

This characteristic of the 2-level phase has even more consequences which any other assignation with more levels will elude. As mentioned before, the signal will be either filtered or “folded” in frequency by the aliasing effect in discrete signals. This means that in the end, it will not have very high frequencies. If it is assumed that the aliasing effect has little effect or either the signal is filtered, in the points of change of phase there will be something like the mean of the values in one direction or the other. The problem is, the mean angle of a signal with a phase of $\pm \pi/2$ values will always have a phase of $\pm \pi/2$. This is: The mean value of a two leveled phase of this kind will lead always to a two leveled value, as it can be clearly seen making the mean of a real value, which will always be real.

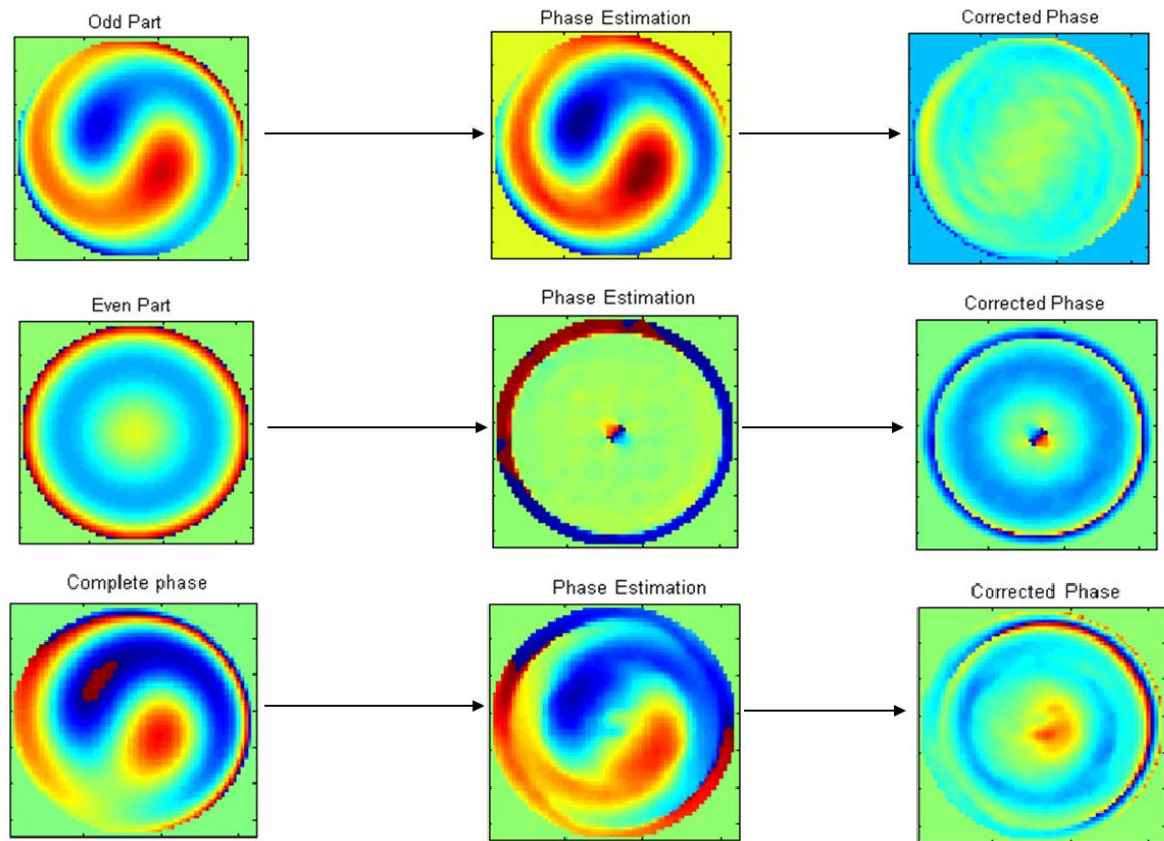


Figure 3.4.2: Two levels approximation of a simplified odd (above) even (middle) and complete (below) phase. In the right column, the input phases are shown. In the middle one are represented the estimation made by the algorithm, and in the left one are the results of subtracting this estimation to the real input.

In other assignments, the phase will make a slope in the discontinuities, which, if the signal is correctly predicted, is easy to see that will make a better prediction of the more or less smooth signal it approximates than a signal with phase jumps.

It also means that any discontinuity will have to be compensated only by the amplitude. So if there is a change from $\pi/2$ to $-\pi/2$, at some point in between the amplitude will be made zero or near zero, factor that can distort greatly the signal, as can be seen in (Figure 3.4.3), in which we can also see some aliasing in the assigned phase.

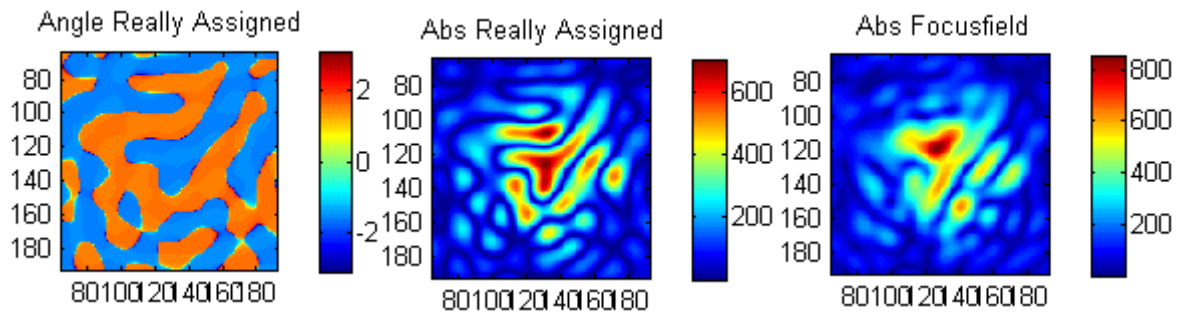


Fig 3.4.3: In the left, repercussions of having aliasing in the aperture visible in the focus phase. In the right, real amplitude in the focus. And in the center, filtered amplitude in the focus.

3.4.2. Division in constant phase zones

As previously said, both to simplify the focus and to group zones with similar characteristics, a watershed algorithm is applied to the intensity at the focus. The zones corresponding to each maximum of the focus speckle pattern can be separated and treated independently.

This part of the algorithm is probably the one which is more difficult to model, because it makes use of an Image Processing tool which is difficult to model in terms of physical characteristics. However, visually it's easy to see that it makes a pretty good job separating zones in most of the cases.

To perform this approximation, the negative of the intensity in the focus is supplied to a watershed algorithm, which returns the separated zones. After that, the phase of each pixel of the focus is weighted by the intensity of this point. Then, the weighted phases of each zone are added, and divided by the total sum of the intensities in that zone. After that, this remaining phase is treated as in the previous part, and the sign of it determines the final phase of the zone.

The weighting part may seem superfluous, and if the algorithm is working correctly it will in fact be so. The problem is this will not always be the case, and it is possible to some zones to **merge** if the maximum of one of them is high enough and/or the decreasing slope of them is quite smooth. If that is the case, it is possible to have in the same zone a big surface with small values and one sign, and a smaller surface with much higher intensity and different sign. In that case, if no weighting is done, the

sign would be very probably the one of the bigger surface, even if its intensity is always near zero. If the total power of the other merged zone is small, this is not a big mistake. But if it has much more power than the bigger one, the algorithm would be making a huge error by assigning the phase without weighting. If this weighting is done, the high power of the second zone will compensate the low power of the other one, and the phase will be assigned with a more logic criterion.

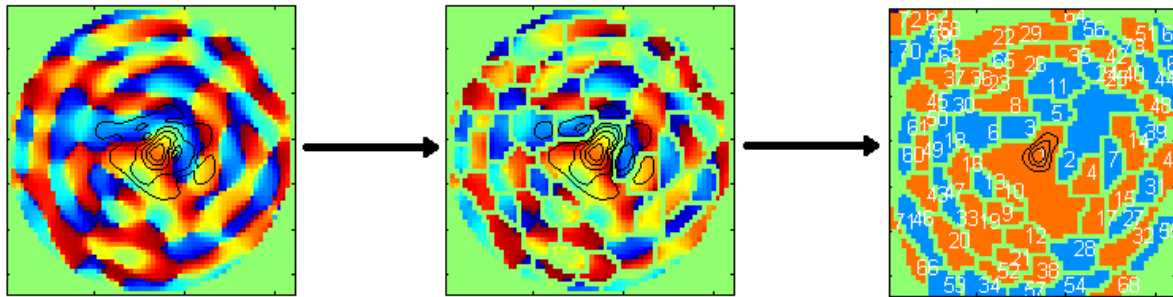


Figure 3.4.4: Process of creating a segmented version of the 2 level evaluation of the phase

In (Figure 3.4.4) is possible to see the previously mentioned way of segmenting. The first image represents the focal intensity with the black level lines, and the focal phase with the colors. In the second image, is possible to see the different regions that the algorithm has determined, divided by light green lines. And, in the third image, this zones have been ordered in terms of the intensity value of its maximum (white numbers) and the phase has been leveled either to $-\pi/2$ (blue), or $+\pi/2$ (orange). In the final version the light green lines would not be present when making the phase assignation, but they have been uncorrected here to make easier the visualization.

Another factor to consider is the symmetry of the phase in the aperture, which also have a very particular effect in this part. As it has been seen in Chapter 3.3, a pure odd signal will be completely real, which will mean that all the changes between signs of the phase will be at minimums of intensity. If the behavior of the watershed algorithm is analyzed, it's easy to see that this implies that a perfect zone approximation will be done in this case. On the contrary, for an even phase it is possible for the correlation intensity/phase to be so low, that the transition makes no

appreciable changes in the slope of the intensity, making it impossible to the watershed algorithm to make a separation of zones there.

3.4.3. *Treatment of only High Power Zones*

This part is the one that has a smaller effect in the overall system, if the aperture is small enough. In the previous part, a maximum number of 300 zones were treated. This is not generally feasible, because like it has been told before it makes the system run very slowly. Furthermore, an error in the prediction of a single zone, by cascade effect, distort the results of a big number of zones, and this effect is worse the more separated to the center the zones are. Therefore, a limitation in the number of zones to consider has been implemented. Experimenting with the disposable data, considering less than 75% of the power distorted the signal in many cases, so the optimum values have been set at a minimum of 80% of the power treated. However, for very spread fields this could still lead to a very slow algorithm, so the maximum number of zones has been limited to 50, a high value but still a feasible one.

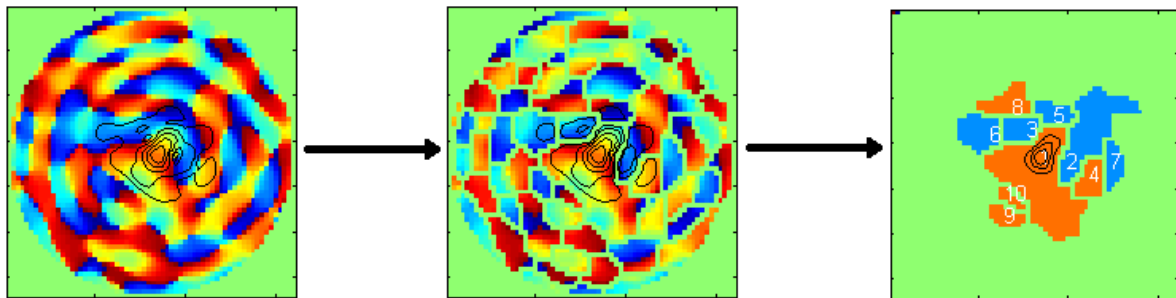


Figure 3.4.4: Limitation of the number of zones to treat by the algorithm

In figure 3.4.4 can be seen the same effect as in the previous example, the segmentation tool in action. However, in this case it has been determined that ten zones are enough to make a good approximation, making the algorithm much faster.

Like it will be seen in the next chapter in Figure 4.8, the distortion this effect produces is very limited, while the improvement in the computational times is of the order of 10 times faster.

3.4.4. *Numerical analysis of the assumptions*

To evaluate the impact this assumptions have in the correction of the system, several series of simulations are run with each of their conditions. These simulations are made, as told before, with artificially distorted fields generated with PiLab, with different conditions for elevation angles 10,20,30 and 90, and aperture sizes from 0.4 to 1 in jumps of 0.2. With this, is possible to obtain an acceptable number of r_0/D values of between 0.05 and 2, and some values outside this margin that are excluded from the analysis. Once the simulations are made, a CDF of the SR is represented for the inputs and the 3 outputs.

In a CDF, the x-axis represents the values used for the representation (in this case, the SR), and the y-axis represents the probability of being under each x-value. This means that if we look at the x value in which the y is equal to 0.1, the 10% of the values will be under it. Similarly, for a 0.5 the percentage will be 50%, and for 1 will be 100%. It is a very useful tool to visualize values distributions.

As can be seen in Figure 3.4.5, the curve regarding the input values (labeled “Non corrected” and represented in black) has 90% of the values above 0.1, the mean at 0.2, and only a 10% of the values are already good enough for having an input SR of above 0.55. Of all the curves, it is the one that grows more slowly, showing that the input SR are more or less homogeneously distributed between this values.

Making the 2-level phase assumption (named “leveled known phase”, in color red) takes this homogeneity away. Most of the values after making a correction with this assumption are concentrated between 0.4 and 0.65, with the mean in 0.5. This means that, if this correction was possible, it would not be perfect but would have a very uniform output of around 0.4 SR. And when talking about communications, is better to have a moderate uniform signal than one with high peak values but high fluctuations. A constant moderate level can require higher amplification, but a variable level would lead to loss of information in one direction and saturation in the other.

Segmenting (“With segmented regions”, in blue) has a non-negligible effect in the fields, as the highly compressed 2-level curve derives to a curve between 0.27 and 0.63, and the mean value is reduced from 0.5 to 0.4. It still has quite acceptable values, and almost no value below 0.2, but some alternatives to the direct segmentation are still feasible. In the future, they should be considered to get the curve as near to the previous one as possible.

And last, the neglect of lower power zones has a hardly noticeable effect (“Only high power zones”, in yellow), which leads to think that, when a definitive algorithm is developed, some speed-up could still be made here by further lowering the maximum number of zones to consider.

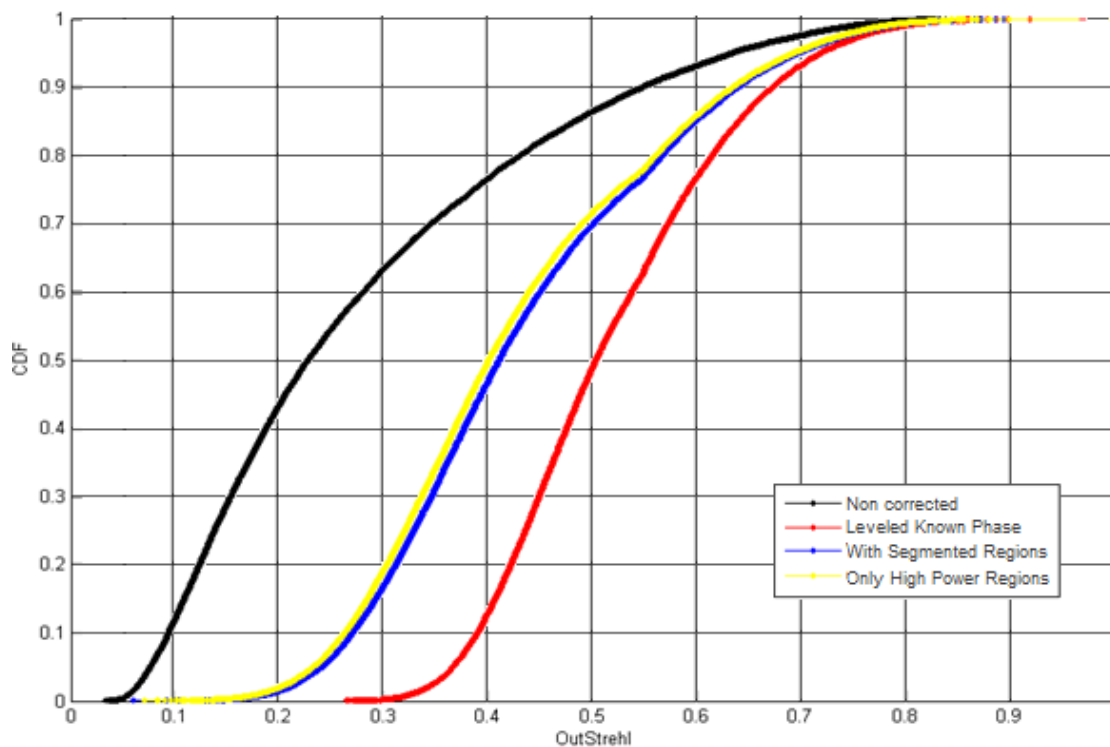


Figure 3.4.5: CDF of the evaluation of the different assumptions, in relation to the input

3.5. Implementation of the Phase Prediction

3.5.1. *Directly observable Relations*

All the previous considerations were either automatic or with not much space for variations inside the algorithm, because they are the ground truth that must be assumed in order for the rest of the algorithm to work. Also, they can be clearly observable, and without further development can be analyzed once the environment is created. This correlation between the intensity in different speckle zones and the relation of their phases must be determined manually, and is not so easy to see the big picture.

To evaluate the different implementations, the rule that will be used will be this evaluation of the correctness of the assignments. This is mainly because the ideal phase of each field is already stored, and to make this evaluation it is only necessary to compare with these values, being much faster than making the whole cycle of simulations, which can last for many hours, or even days if one want to be thorough. Also, it has been empirically seen that these two characteristics are very closely correlated with the SR improvement. The power with correct sign has the advantage to the simple percentage of correct assignments of pondering with heavier weights the peaks of high power, making this ratio even more correlated with the SR.

Also, a division in the input fields is made, to make it clearer the quality of a implementation and, if necessary, to make possible the use of different implementation depending of the characteristics of the input field. The characteristic used to make that division is the r_0 , for values under 0.2, over 0.5, and between this two values. To make them simple to refer, the lower values of r_0 will be known as B (or bad) fields, the higher as G (or good) fields, and the ones in between M (or medium) fields.

The first relations directly observable (already stated in [GIG10]) can be seen as an example in image 3.5.1. When two peaks have both a considerable intensity and a high step between them, it is in almost any case a sign of a change of the sign of

the phase between them. This general idea can be stated in many different ways, and so it has been made.

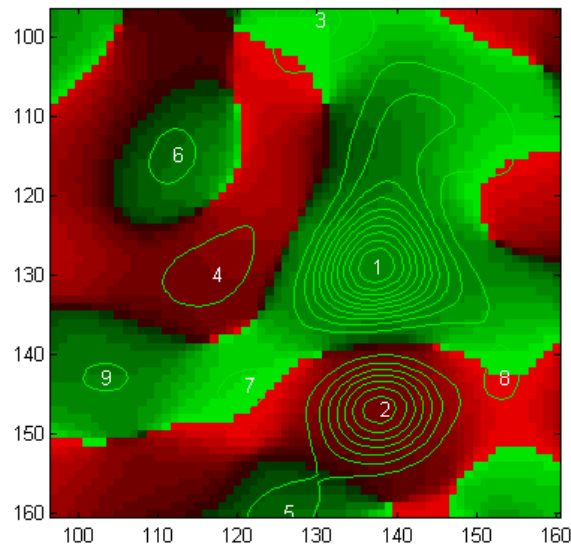


Figure 3.5.1: Visual observation of the relation between phase ($-\pi$ to 0 with dark to bright red, 0 to π with bright to dark green) and the intensity speckles (green level lines, with numbers in the peaks corresponding to the order in terms of magnitude)

Mathematically, most of the metrics that can be imagined following that principle are based in the characteristics of the direct path from one peak to another, which can be obtained evaluating the function in a straight line between the two peaks. It is easy in that case to calculate the minimum, and see the difference between the peaks and this value. It is also possible to calculate the mean and the variance, two characteristics very useful to concentrate the information of the magnitude of the peaks and if there are very big variations only in two variables. And even some less intuitive but quite accurate metrics, like putting a threshold of amplitude and seeing if the signal goes below it in its way from one peak to the other.

To evaluate this metrics, a single value must be obtained, and then a decision is made. For the last option is direct, it goes or does not go under the threshold. From the variance and the mean, if the variance is not normalized it must be divided by the square of the mean, and from the minimums and maximums is easy to make a simple mean of the two maximums, and divide it by the minimum.

Each of this metrics (and other similar ones, but less representative) have the characteristic of having a relatively low computational cost, some of them less than the others. Calculating the mean and the variance of the whole line is in general the most requiring of the techniques described, but even in this case it is a cost easily

assumable, and even more if this time is consider in relation to the bottlenecks of the system, like for example the watershed algorithm.

However, there is another point to consider when working with this kind of metric: There must be a threshold for the value we obtain with it, to state the point in which we change from changing the phase to assigning the same phase. In fact the experience tells that it is better to have two thresholds to help differentiate the points in which the decision is not clear, at least with the metric used. With this, it is possible to entwine different metrics to complement each of them with the information of the others in the cases when there is an indetermination in any of them.

For this metrics, the most important point of its evaluation is to correctly assign the best threshold (or thresholds) for making the prediction. For getting these values, when working with only one metric, a simple iterative method can be used, and getting by try and error the threshold that maximizes the number of correct assignments, or the one that maximizes the power correctly assigned.

The problem, though, is that all these techniques rarely say that the phase should change when it should not, but when they predict that it should not there is a big chance of being wrong. In numbers, only a 65% of the values are correctly predicted, going up to 75% if only the G fields are evaluated.

This is because the observable property in which they are based almost always implies a change of phase, put there are many others that have not been seen that also lead to this change of phase. This could be corrected with a composition of metrics, but the problem in this case is that all the metrics that have been tested are based on the same observed characteristic, which means that they have a big correlation between them and makes them difficult to complement each other.

3.5.2. *Machine Learning Relations*

. As it has been seen before, the direct observations does not lead to a clear relation, and so it appears the necessity of creating some automatic way of determining them. This is the point where the Machine Learning algorithms become an interesting part of the project.

There are many machine learning algorithms, but most of them are only able to output a relation in very simple terms, like the sum of pondered inputs. This is still very useful, because it could be used to automatically making the composition of different metrics, for example, but it is not so when this simple relations have already been tried manually and discarded. The ones that are particularly interesting in this case are the ones that are able to extract more complex relations, like the Neural Networks.

Still, for any machine learning algorithm there are some characteristics that must be determined from the start, which are the inputs and the outputs. For each of them, some decisions must be made.

Regarding the outputs, they must be the values assigned to the phase, translated to a simple binary decision. It has been chosen for the system to assign the value one if the phase has a different sign than the phase of the maximum, and zero if it is equal. So for each field there will be at the start a zero output for the first zone (the one with higher power), and undetermined values in the others.

The inputs, on the other hand, are not so clear, and its determination is a very important part in the creation of any ANN. The characteristics that seem to have a relation with the phases of the different zones must be chosen manually, with the subsequent possibility of something important not being considered. This is the reason why the best approach is to make a compendium of all the characteristics that may have an impact in the decision, and train the system with them. This will obviously lead to a very slow training and treatment of the signal in the ANN, but this can be solved afterwards. Looking directly at the parameters the training obtains, it's relatively easy to see if any of them is not correlated with the decision that the network does, because its weights will be very low. However, it must be taken into account the fact that to be able to compare the weights, and also in order to speed

up the convergence of the algorithm, it is necessary to make a normalization of the inputs. This normalization can be easily made by subtracting the mean and dividing by the standard deviation of the values in the training examples.

<i>Peak Number, sorting the peaks by intensity value of the maximum</i>
<i>Diameter of the aperture, in meters</i>
<i>Area of the zone of the first peak, normalized by the square of the aperture</i>
<i>Area of the 4 nearer zones, including the one evaluated, normalized</i>
<i>Power of the zone corresponding to the first peak</i>
<i>Power of the 4 nearer zones, normalized by the power of the first</i>
<i>Minimum of the line connecting the first peak with the evaluated, normalized</i>
<i>Minimum of line connecting the evaluated peak to the 3 nearest, normalized</i>
<i>Mean of the line connecting the first peak with the evaluated, normalized</i>
<i>Mean of the line connecting the evaluated peak to the 3 nearest, normalized</i>
<i>Distance to the first peak normalized by the aperture diameter</i>
<i>Distances to the 3 nearest peaks, normalized by the aperture</i>
<i>x and y distances to the first peak, normalized by the aperture</i>
<i>x and y distances to the 3 nearest peaks, normalized by the aperture</i>
<i>Peak intensity values of the 4 nearest peaks, including the one evaluated</i>
<i>Phase that an ideal airy distribution would have in each zone</i>
<i>Phase of the 3 nearest peaks, if already known</i>

Table 3.5.1: Neural Network parameters

Once all the inputs and the outputs are obtained, another decision must be made: How to organize them. It has been previously stated the fact that the number of zones considered in each field should be dynamic, so the application of a single static neural network to determine all the phases at a time becomes difficult. The only way to make it so would be making the system for the greatest number of possible zones, and introducing values that don't distort the signal in the cases when

the number is inferior. However, doing so could make not every zone to be treated the same way, as the further away zones will have fewer values to be trained with. This is the reason why an approach of applying the neural network independently for each peak has been chosen. With this approach, the total number of training examples is incremented to the sum of the total number of zones in each field, in front of a maximum of the total number of fields with the other method. The organization of the input layer is represented in Table 5.1.

Once this general structure is defined, it is also necessary to determine the data that will be used as training data, and the one that will be used to evaluate the system afterwards. Like it has been stated before, with the data at our disposal from PiLab, different sets of 1000 simulations are made with apertures from 0.4 to 1 and incidence angles of 10, 20, 30, and 90. This means that the data to process is composed by 28k fields, each of them divided in at least 10 peaks, and each peak with near 40 input parameters. The general assignation in this cases consist, if not using cross-correlation data, in assigning the 70% of the data to the training, and the remaining 30% to the evaluation. However, due to the long time it will consume and to the fact that not so much data is necessary, a 30% for each one is used instead. To make sure that this data is not correlated, the fields for each group are taken randomly from the whole set. Even if the training is repeated for further improving the parameters (which may happen if the convergence is too slow and must be cut before the end), the training data will change, preventing cases of randomly picking some similar fields by chance.

The rest of the decisions to be made when creating an ANN are not so easily determined from the data. To begin with, it must be determined how many hidden layers are necessary, the number of nodes that each of them will have, and the parameter λ to be applied during the training. All of them can be determined as try an error, but there are some general rules in order to make this decision. More hidden layers and more nodes mean possibility of more complex relations. However, it also means increasing greatly the time it will take to train the network and to use it. Also, it creates the possibility of overfitting the training data, in which case it must be corrected increasing the λ parameter. In the case discussed in this document, an ANN with 1 hidden layer, $\lambda=3$, and with a number of nodes in the hidden layer equal to twice the inputs, the results are already acceptable. With both the training data

and the evaluation one, the results are of around 80% of correct values and 85% of the power, with values as high as 85% correct data and 90% of the power in the G fields.

However, these results are a little different when applied in the algorithm in real time, mainly because of the use of the phase of the 3 nearest peaks in it. This factor leads to two different effects in the evaluation of the data which cause a distortion in the results.

On one hand, the algorithm only knows the phase of the already evaluated peaks. This means that when it is evaluating the second peak, the only phase information it possess is the one regarding the first peak. Obviously it can still evaluate the rest of the data of every other peak, but without the phase information in most of the cases this is not very useful. This can be corrected doing a first iteration, estimating the phase of each peak, and then doing a second iteration with the phase signs obtained this way. It would not be perfect, because making this second iteration would mean working with the data of the first, which can be wrong, and so a third iteration would be needed, and so on until no changes are made. However, this can be very time consuming and it's possible to enter in a loop in which there are two or more phases which change in every iteration, making it impossible to reach a stationary. For this reason, the options more recommended are making a single determination without part of the data, or doing a single second iteration with the data of the first, if the time constrains permit it. The data presented in the following parts is obtained with the first option, although with little modifications the second one can be achieved.

The second problem, however, is not so easy to correct, although is very related with it. As it has been seen before, in the training data the true phases of each zone is used as input to determine the one of all the neighboring peaks. This means that for every peak the determination of the phase assumes that all the other phases are already correct. In the case of having high concentrated powers, like in the G fields, this is not a big problem, because only ten zones will be used, and also the probability of making a mistake in the prediction is very low. But in the B fields is normal to have 50 zones, and even in the ideal case one of each 5 will probably be predicted wrong, as seen before. This means that all the zones touching this wrong

zones will be making it's phase determination with data that is not correct, and so it is highly possible that They also make a mistake in the prediction. This can lead to a chain effect in which the more far away from the center the peak is, the most undetermined the correctness of its phase will be. To partially correct this, it is possible to make a training iterative method, which takes in each loop as inputs not the real values but the values of the previous iteration. The advantage with this correction with respect to the previous one is that it only increases the time needed to train the ANN, not the execution time, and because of that it should be implemented in the future.

Another solution that may lay between would be the creation of two different neural networks. The first one, is trained for each zone only with the ideal phase data of the neighboring ones that are already assigned, and the second one is trained with the outputs of the first one in the real case. With this, each peak has the data of the phase of all the neighboring ones, and it takes into account the fact that the inputs are not the real values.

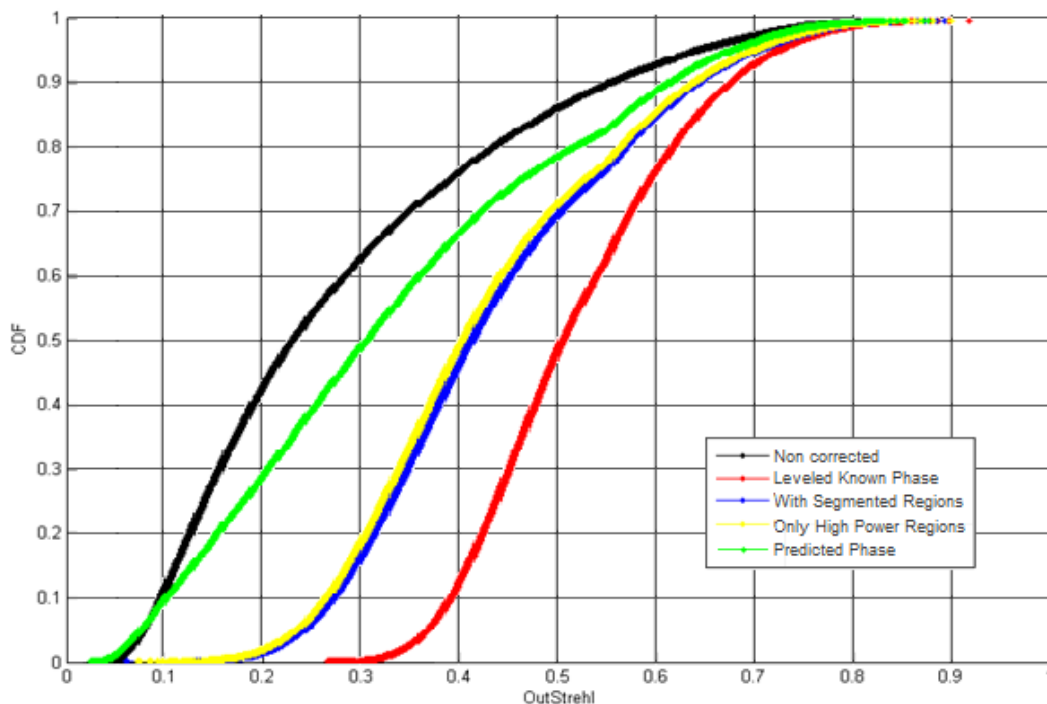


Figure 3.5.2: CDF of the evaluation of the different assumptions, in relation to the input and the ANN implementation

In this project, because of the time restrictions, only the first general solution has been tested, although the generation of the other models can be easily achieved in a reasonable time if necessary.

But even with the problems mentioned above, the behavior of the system continues being relatively good (Figure 3.5.2), and improves very much the ones obtained with the algorithms extracted from directly observable relations. Looking at the graphic, the worse fields (below 0.1 SR) are not improved, but in the worse fields is in the ones that the previously mentioned problems are more evident. Because of that, the future implementation of any of the above modifications to the ANN should make a big shift in this zone, and a moderate one in the rest.

One of the most interesting conclusions is that even with the possibility to make these considerable improvements, the system already shows signs of being quite accurate. So the ANN should be considered to be used in this algorithm implementation and many other related topics, even if it is later changed for a less complex implementation.

4. Conclusions and Future Lines of Work

Nowadays, Optics is one of the fields growing faster, with new applications being born each day. In particular, the free-space Optics is an interesting way of solving many of the classic problems with the radio channels, as the limited throughput.

However, these communications suffer more intensely from the atmospheric turbulences. The objective would be to couple the signal to a single mode fiber and it is not possible to do so with an optical signal so distorted as the one resulting from a normal Satellite-to-Ground communications. This generates the need of an Adaptive Optics system. These systems, unlike the free space optical communications, have already been used for a long time, as they are a key part in astronomy. Systems with wave-front sensor and deformable mirror have been extensively analyzed for that ambit.

The problem is that the astronomy systems are made for working in some very particular situations. They work well in astronomic locations and at higher elevation angles, because in these environments the atmospheric distortions are relatively low. But they are not designed for standard locations, low elevation angles, and in general strong turbulences. Furthermore, the wave-front sensors which are the key of the astronomy systems are expensive and very sensible to calibration.

From there appears the need for AO systems specifically developed for communications, like the one working with the APHEFIS (Adaptive Optics by Phase Estimation from Focal Intensity Speckles) algorithm. The basic concept behind this is to work with a software algorithm to detect the atmospheric distortions. It analyzes the intensity data in the focus, approximates the phase with the information contained in it, and uses the inverse Fourier Transform to counter the focusing effect of the lens, obtaining this way an approximation of the desired aperture phase. With this, the complexity added by the wave-front sensor is eliminated.

For doing this, an environment in Matlab has been created to simulate an adaptive optics receiver system to work with the APHEFIS algorithm. For this environment, a GUI has been implemented, which could be easily modified to add other parameters to the system. Also, inside this environment the analysis of different characteristics has been implemented and it has all been integrated with the complex fields generated by PiLab, a Toolbox of the DLR. This way, it is possible to analyze the performance of this or any other future AO system based in software.

Once this has been done, the algorithm has been deeply evaluated. It has been decomposed in the different ideas it introduces: there is a correlation between phase and intensity in the focus, a 2-level approximation of the phase in the focus is good enough (Strehl ratios improved by 0.3 points in the worse fields), it is possible to make a segmentation in constant phase zones (the improvement is reduced to 0.2 points, still considerable), and the treatment of only the high power zones does not deteriorate appreciably the quality of the estimation. Using the data obtained with PiLab, each of these assumptions has been analyzed and its impact on the system determined, showing a big potential to improve the optical field coupling quality.

To implement the determination of the phase of the zones, systems using direct relations have been tested, with results way under the desired. After that, a system with an Artificial Neural Network has been developed. It has been created and adapted to work with the specifics of the system, letting it prepared for being easily modified and operated. With this, the system can be entirely operated through the Matlab GUI, only requiring code modifications for deep changes.

The ANN system has been trained using the simulated fields under several turbulence conditions. The first results have been achieved, showing an improvement in more than 93% of the cases.

With the intention to further understand the behavior of the system in order to improve it, an analysis of the Zernike Polynomials has been made. Testing the system with different phase distortions extracted from these Polynomials has shown a direct correlation between the correction capacity of the system and the parity of the field phase in the aperture. It has been seen that, due to the particular properties of the field phases and of the Fourier Transform, fields with odd aperture phases can have an almost perfect correction. On the contrary, even field aperture phases make

the algorithm introduce discontinuities which are difficult to surpass. In this direction, the theory analysis has been started, but further analysis would be required.

To finalize, several lines of work should be further investigated for a better characterization and understanding of the algorithm. The most important ones are the following:

- **Relate the aperture phase symmetries with the focus image pattern, and development of more theory insight:** The relation between the aperture phase symmetries and the behavior of the system has been clearly observed in chapter 3.3. Therefore, relating these phase symmetries to focal intensity patterns will greatly improve the algorithm.
- **Variation of the watershed algorithm for a better zone determination:** The analysis has revealed that an ideal assignation with watershed zones already can do a reasonable improvement. However, in some cases the extracted information is not correct, leading the algorithm to not work properly. The most problematic cases can be identified and treated separately if the computational cost is acceptable.
- **Improvement of the ANN:** The Artificial Neural Network has been a very useful tool, but the system could be further improved with some of the possibilities already stated in chapter 3.5.2 and with new NN technics that appear each day.
- **Computational time analysis:** Matlab is a very time consuming language, making it not adequate to test the time efficiency of an algorithm. In this direction, the implementation of a C, C++, or similar lower level language will be necessary at some point.
- **Continuous time version of the algorithm:** It has been seen that the algorithm is feasible for static images, but it has for a dynamic environment.

5. Appendix

Appendix A: Acronyms

ANN	Artificial Neural Network
AO	Adaptive Optics
APHEFIS	Adaptive Optics by Phase Estimation from Focal Intensity Speckles
CDF	Cumulative Density Function
D	Aperture Diameter
DFT	Discrete Fourier Transform
DLR	Deutsches Zentrum für Luft- und Raumfahrt
SGL	Satellite-to-Ground communication link
f	Focal Length
FFT	Fast Fourier Transform
FSP	Focal Speckle Pattern
GUI	Graphic User Interface
H	Heterodyne Efficiency
IFT	Inverse Fourier Transform
LEO	Low Earth Orbit
r_0	Fried Parameter
RF	Radiofrequency
SHS	Shack-Hartman Wavefront Sensor
SR	Strehl Ratio

Appendix B: Files index

Created

APHEFIS_GUI.m/APHEFIS_GUI.fig:

Support GUI for dealing with the simulations, calculations, and graphic representations. A guide of how to use it is provided in “Appendix C: GUI guide”. It shows the data through results_table.fig

APHEFIS_bridge:

In charge of connecting the data to the algorithm for its correction and return the results. Also in charge of the images, due to the fact that the information required to represent them is only present in there.

APHEFIS_CORRECTION.m:

Actually applies the algorithm. Given an intensity focal distribution, it makes the watershed, extracts the characteristics of the regions, and applies the NN to it.

APHEFIS_plot.m:

Support function (not used by the GUI) usefull for making graphic representations from analyzed fields.

APHEFIS_cut.m:

Because the information extracted from the fields is stored in a structure with different tables, accessing to a particular data is not always trivial. This function returns a structure with only the fields specifieds.

APHEFIS_search.m:

Looks in the structure for fields with similar characteristics to the input ones, and returns them. It's possible to control how similar each characteristic

should be, and how many characteristics must be in this range to consider the whole field similar.

APHEFIS_characteristics.m:

Extracts useful information from the file/structure passed. This information include means and variances of the most relevant parameters.

neural_network.m:

Core of the NN used. It is not an optimized version, and for time calculations one with less computational cost should be adapted and used instead.

cut_image.m:

Very simple function used for extracting a part of a bigger image without using the image toolbox.

optimize_metric.m:

From some initial data, it extracts the characteristics of the optimum NN for fitting to them. Use the auxiliary functions `nnCostFunction.m`, `sigmoid.m`, `sigmoidGradient.m`, `featureNormalize.m`, `fmincg.m`, `gradientDescentMulti.m`, and `randInitializeWeights.m`.

Zernike_cicle.m:

Calculate the Zernike coefficients of the fields stored in the file or structure passed to the function. It requires the phase of these fields to be stored previously. It uses some auxiliary functions as `MatchZernike.m`, `oddvec.m`, `zernimat.m`, `distmat.m`, `zerninum.m`, `angmat.m`

static_metrics_statistics.m:

From the data received from `APHEFIS_bridge`, extracts some additional data, as for example the improvements from dividing an output value from the input one.

polartrans_mod.m:

Transform an image to polar coordinates.

peaks_xy_dg_mod.m:

Search the peaks in an image, and returns its values and the x,y coordinates where they are located..

Metric1-5.m:

Old metrics currently not in use because of the NN. However, they are still present in case of necessity.

transformtitle.m:

Auxiliar function used for pasing from a variable name to a more displayable name. Mostly used for graphics representation

neural_network.m:

Core of the Neural Network used. It is not optimized, and for time calculations one with less computational cost should be used instead.

neural_network.m:

Core of the Neural Network used. It is not optimized, and for time calculations one with less computational cost should be used instead.

Adapted or not modified:

GoldsteinUnwrap2D.m:

Function used for making an unwrapping of the phase. It uses several auxiliary functions: FloodFill.m, BranchCuts.m, PhaseResidues.m

focustransformation.m:

Transform an aperture field into the focus plane. The inverse transformation is done by backtransformation.m

apmask_v2.m:

Creates a circular mask of the specified size.

COGpixels.m:

Calculates the center of gravity of a field.

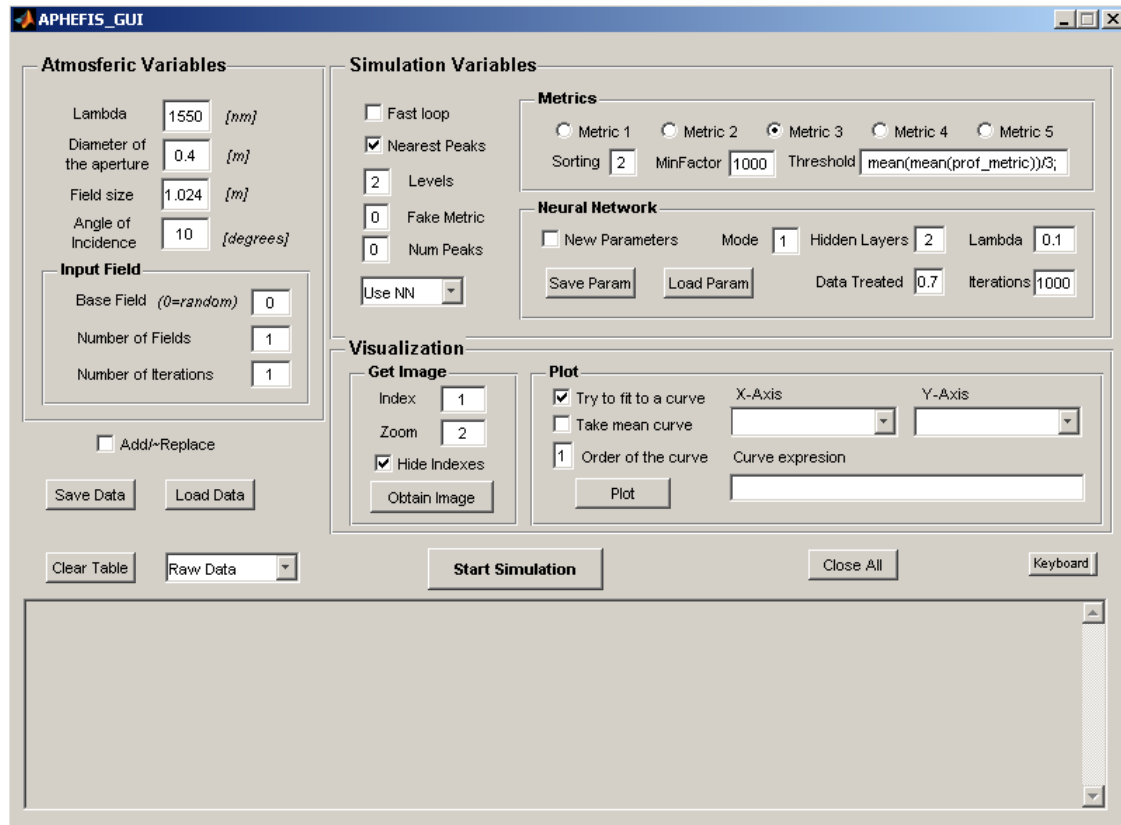
HE_Aperture.m:

Calculates the HE of a field from the values of it in the aperture plane.

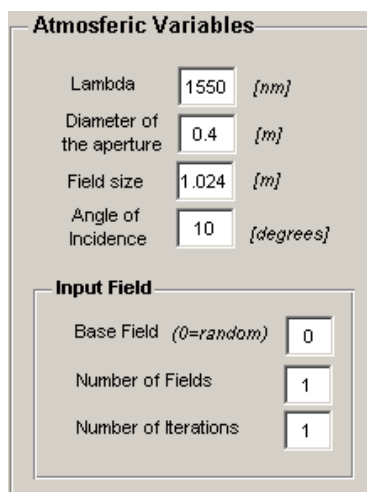
stadistics_ifield.m:

Gives useful information about a particular aperture field. In particular, it returns the r_0 and the number of branchpoints that are present in the aperture. Uses the auxiliary functions *r0_getstructf_Fried.m*, *branchpoints_function.m*, and *calculate_rho_int.m*.

Appendix C: GUI guide

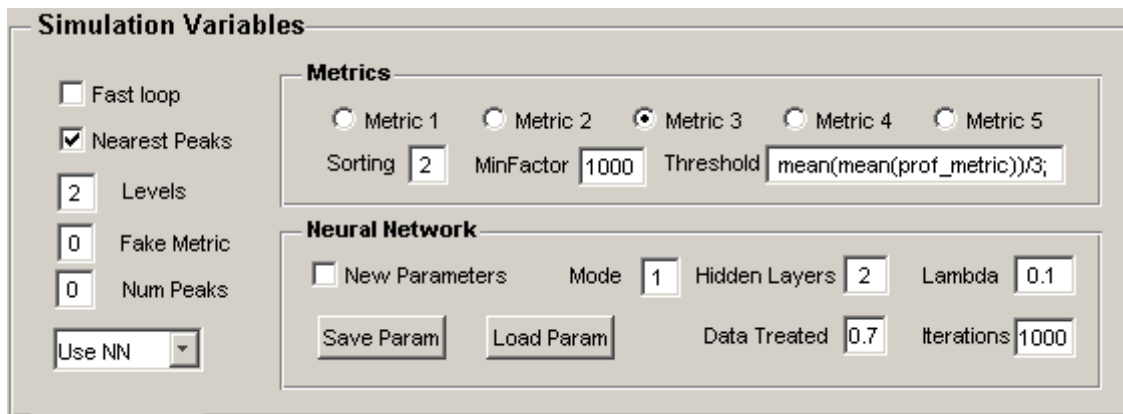


This is how, at the moment, the GUI realized for developing the project more comfortably looks like. There are different parts which I will explain separately.



This is the part that depends of the atmospheric conditions that we have. In other words, which field/s will be our input/s.

We are limited by the fields that we have, so the alpha is a fixed value, the possible angles of incidence are 5, 10, 20, 30, 90, and we have a total of one thousand fields for each incidence angle.



Simulation Variables

☐ Fast loop
☒ Nearest Peaks
 Levels
 Fake Metric
 Num Peaks
 Use NN ▼

Metrics
☐ Metric 1 ☐ Metric 2 ☒ Metric 3 ☐ Metric 4 ☐ Metric 5
 Sorting MinFactor Threshold

Neural Network
☐ New Parameters Mode Hidden Layers Lambda
 Save Param Load Param Data Treated Iterations

In this second section, we can change the way we want our simulation to behave.

The first thing we have to decide is if we are going to use a directly calculated Metric to Predict the phase of a new field, if we want to do so but with a Neural Network, if we are going to train a neural network with some stored data, or if we want to see the behavior of the actual Neural Network with this stored data. For determining this, the sliding menu in the downside left corner must be selected.

Above this menu, there are some other options that we should consider:

- The “Fast loop” option is used to extract the characteristics of a determined field or fields, but without making any prediction. This is very useful if we want to have the information of this fields to train a Neural Network, because it will spare us a lot of time.
- The “Nearest Peaks” option indicates our system if we want to take the zones in relation to the distance to the maximum, or in relation with the power. If taking a considerable of zones it will approximately be the same, but if this rule is not followed this is not necessarily true.
- The number of peaks can be any number between 10 and 30 for the moment, or zero. If 0, it will be considered a dynamic number of peaks to guarantee that a minimum power is considered.
- If we are using a Fake Metric (look below) we can also change the levels of the zones that we will use, not restricting it to 2.

- The Fake Metric option lets us see how good the correction could be if only some of the limitations of the system are applied, and not all. For all of them, the knowledge of the actual phase is used, and therefore they are no true metrics:
 1. Assign the phase as the watershed system could actually do: one phase per zone.
 2. Assigns directly the leveled phase to the zones we are treating but without restricting itself to one phase per zone.
 3. Assigns the leveled phase to the whole focus field.

If we want to use one of the determined metrics at the disposal of the program, we must fix the values of it with the options above to the right, inside the “Metrics” box.

There are five metrics operating with a two level prediction based in the watershed zones of the phase. For each one of them, we can control if we want to start assigning phases in order of higher value of the peak in it (sorting=1), or by proximity to the maximum (sorting=2). Also, we can decide to not make decisions for zones which peak has less than 1/MinFactor the maximum, because this decisions are less critical and always more ambiguous to make. Last, we can decide which threshold we want to apply to the matrix that we will obtain after applying the metric to each combination of significative zones to decide if we change or not change the sign. If we must make a more than two level prediction, this part should change accordingly.

The five metrics made the following operations for obtaining the value we will use to determine change/not change of sign:

- 1- $(-distance * ((peak_i + peak_j) / 2 - \min(\text{prof_vec})) / \text{mean}(\text{prof_vec}))$
- 2- $(-variance(\text{prof_vec})^{0.5} / \text{mean}(\text{prof_vec}))$
- 3- $(\sum(\text{abs}(\text{lin_fit} - \text{prof_vec})) / \text{mean}(\text{prof_vec}))$
- 4- $(-\min(\text{prof_vec}) / ((peak_i + peak_j)))$
- 5- $(-2 / (\sum(\text{prof_vec} < \text{maxpeak} / 20) * (peak_i + peak_j)))$

Legend: *distance*=distance between peak of zone i and peak of zone j.
prof_vec= vector with the values of the intensity in the straight line between the peaks.
lin_fit= direct line (polynomial of order 1) between the peaks.

For using, applying, or training a Neural Network, we can use the options in the box named with this label.

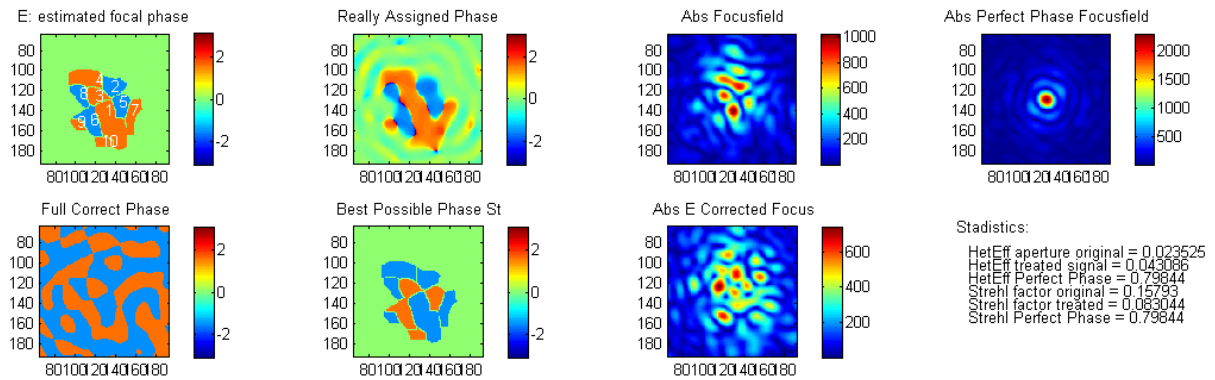
In the case of using the Neural Network on a new field or applying it to a set of fields already analyzed, we can charge the parameters with the “Load Param” button.

If we want to train new (or not) parameters, we should look at the other options.

- The “New Parameters” option, will determine if we use the parameters stored in memory (if they exist) are used as starter parameters to continue training, or if for the contrary we want to start with new ones.
- The “Mode” will determine the parameters that will serve as inputs and outputs for the network. They are directly inside the code, so only the options that have been considered worthy can be selected. *(Should make a list of what each mode uses)*
- The number of Hidden Layers will determine the complexity of our system. The more hidden layers, the more complex operations the network can do, but the more computational cost it will have (Look at the “Neural Networks” chapter for more information).
- The Lambda serves to avoid our system to overfit the training set, which means that it would work good for it but not for other inputs. A high lambda implies a low risk of overfitting, but will also make less likely for our system to reach the optimum solution. Warning: This Lambda should not be confused with the wavelength
- We can determine how much data we want to use for training the system. Much data will make the algorithm very slow, while considering a small group of examples will fail to find the important parameters.
- Last, we can determinate how many iterations we want our system to perform. Ideally, the algorithm will stop automatically when it has converged, but it can take a great amount of time, so we have the option to limit it.
-



This part of the menu is used for obtain images of the fields that we have analyzed, in case the numeric information is not enough for our purposes. For reasons of commodity, each treatment of each field has a different index assigned, so for obtaining the image of this field we only need to put it's index, and not the complex label assigned to it. For obtaining multiple images, it accepts Matlab expressions like "1,2,3", "1:9", etc. The zoom is pretty descriptive, and the "hide index" parameter enables us to avoid seeing the "index" parameter in the tables, in case we want to see only the data. The images that we will obtain depends on if there is a single iteration of the algorithm applied to the field or more. In the first case, the image is as follows:

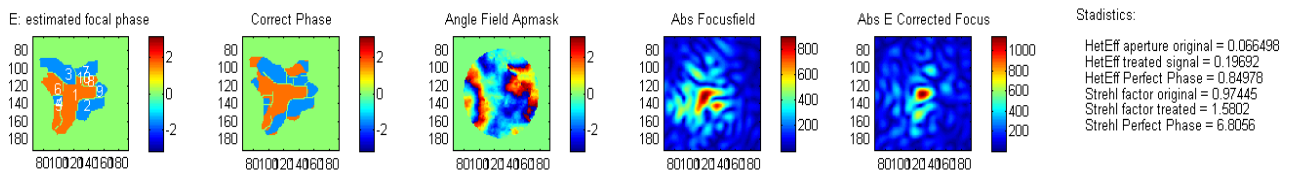


From left to right and from up to down, the images represents:

1. The phase that we have estimated the focal field has. It can be one obtained with a real metric, or with a fake one.
2. The backtransformed and transformed again phase. This is the real assignation we are making, taking into account the anti-aliasing filtering.
3. The amplitude of the focus field. The square of this is the only information that our system has (if not using fake metrics) to make the prediction on 1.
4. The amplitude of the focus field of an aperture field of the same amplitude of ours but with a uniform phase. This is our limit, so it's good to have an idea of how good our system could be at the end.
5. The phase of the full field approximated by the number of levels indicated. This is the phase we will assign if working with fake metric 3.

6. The same as above, but assigning only one phase to each of the treated watershed zones. This is obtained pondering the phase of each pixel by it's intensity, and correspond to fake metric 1.
7. The amplitude of the already corrected field. This is the output we will have from the system, and also the input for the next iterations.
8. Some useful statistics. The images alone can be deceiving and so can the numbers, so it's good to be able to se the two things at the same time.

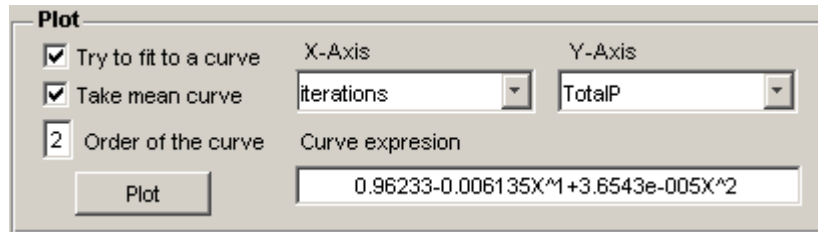
If we ask for a more than one iteration image, for reasons of space and commodity, we spare some images to make more iterations visible at the same time:



From left to right, the images represent:

1. The phase that we have estimated the focal field has. It can be one obtained with a real metric, or with a fake one.
2. The phase of the full field approximated by the number of levels indicated and restricted to the areas that we will be treating. This is the phase we will assign if we work with fake metric 2.
3. The phase of the field at the aperture mask. When doing different iterations, it's useful to see what kind of phase distortions we cannot correct.
4. The amplitude of the focus field. The square of this is the only information that our system has (if not using fake metrics) to make the prediction on 1.
5. The amplitude of the already corrected field. This is the output we will have from the system, and also the input for the next iterations.
6. Some useful statistics. The images alone can be deceiving and so can the numbers, so it's good to be able to see the two things at the same time.

The images we want to represent are easily changed with the cell variable “images” of the system, but I have considered that putting it accessible at the GUI made no sense and made it too much overcharged. However, if it is required it can be easily implemented.



This section enables us to make plots of the data, with any combination of parameters that we have (although some of them making no sense).

The first check-box enables us to decide if we want to try to fit a curve to the data we will be representing. If we do so, we can specify the order of the polynomial we will try to fit, and if we want to make a mean curve approximating the whole package of data. Assuming we do so, in the “curve expression” box we will have the curve expression that approximates it, in case we want it.

For the moment it’s only implemented to work with polynomials of exponents greater than one, which fails to easily adjust to convergent functions, but it can be easily implemented (and it will be so) for other type of functions.



This last sections controls the input and output of data, and another miscellaneous utilities.

The button “Close All” is simply a shortcut which closes all the other opened windows.

Clear table is used to erase all the data actually charged in the memory of the program.

The save button stores the current data in a specified file, and the load data charges it from it. In case we have selected the “Add/~replace” button, the data will be added to the current one in the program, and if not it will replace it.

The keyboard button is used to debug or to make things not currently implemented in the GUI. It opens a command window with the current data of the program accessible, so we can observe it's values, change it, or filter the data. In the future, this button is expected not to be necessary.

The start simulation button is the one that makes the system start computing. We call the function `start_APHEFIS`, which checks the parameters and calls the function `AO_APHEFIS` if they are valid. The input parameters are stored (`handles.cr_input.data`) in a table inside the handles of the GUI, as are the output parameters (`handles.data_t.data`) and some statistics made from them (`handles.static_statistics.data`). We can see any of this tables by selecting it in the left menu of the button. The output is displayed at a new window, and looks like that:



	Index	Strehl Impr	He Impr	He To Perfect	Strehl To Perfect	Power In Zones	Correct Po...	Best Power	R. With Best...	Power With...
F716 1 M3T=mean(mean());	1	0.7751	4.4354	0.1279	0.1281	0.4903	1.3848	0.7251	1.9096	1
F716 2 M3T=mean(mean());	2	0.6039	0.3879	0.0496	0.0774	0.4396	1.5104	0.7672	1.9686	1
F637 1 C350	3	2.7589	19.1442	0.6210	0.6210	0.6362	1.5717	0.9924	1.5836	1.5717
F637 2 C350	4	0.4833	0.4570	0.2838	0.3001	0.7209	1.3870	0.7238	1.9162	1.3870
F995C350	5	1.7308	2.6070	0.5358	0.5358	0.6647	1.5044	0.8495	1.7708	1.5044
F948 M3T=mean(mean());	6	0.7136	1.8338	0.1628	0.1632	0.4273	1.6835	0.9574	1.7583	1

The format of the labels in each row has a common part, which depends on the field and the iterations,

Field*fieldnumber**iteration*

and another that is different depending if we are using or not a fake metric.

F716|1| M3T=mean(mean()); **M***metric***T=Threshold**

F637|1|C350 **C***fake_metric***S***Substracting_phase_of_maximum*

In case of having only one iteration, the *|it|* part is eliminated,

F995C350 **F948 M3T=mean(mean());**

The first column in each of the tables, if the option “Hide Indexes” is not selected, it's the index. The next ones are:

RAW DATA (data_t)

TotalP: Normalized potency in the whole considered focus area. This area depends on the zeropad factor we use when making the transformation.

ZonesP: Normalized potency in the zones we will be treating. Depending on the algorithm, it can be the whole area or less.

AssignedP: Normalized potency in the zones we will be treating in which we make a decision. Some algorithms have the option not to make a decision on a zone if they are not sure of the result or if the zone has little potency, for example.

CorrectP: Power with phase correctly assigned (considering ideal the phase the leveled function) inside the treated area.

BestAssignP: Power that will correctly assign the fake metric2, it is the ideal one restricted to one phase per watershed zone.

InStrehl: Strehl factor of the input field.

OutStrehl: Strehl factor of the output field.

InHE: Heterodyne efficiency of the input field.

OutHE: Heterodyne efficiency of the output field.

PerfectHE: Heterodyne efficiency of a field with the same amplitude at the aperture but plane phase.

PerfectStrehl: Strehl factor of a field with the same amplitude at the aperture but plane phase.

MaximumFase: Absolute phase of the maximum.

Branchpoints Amp: Number of branchpoints in the aperture.

Branchpoints CSM: Number of branchpoints in the aperture.

r0: Fried parameter.

Static Statistics (static_statistics)

Power In Zones: Power in the watershed zones divided by total power in the focus.

Power With Sign: Power with sign assigned divided by power in the watershed zones.

Correct Power: Power with sign correctly assigned divided by power in the watershed zones.

Best Power: Power of the best possible prediction with sign correctly assigned divided by power in the watershed zones.

R Whith Best Power: Potency that will correctly assign the fake metric2, it is the ideal one restricted to one phase per watershed zone.

Strehl Impr: Output Strehl ratio divided by input Strehl ratio.

HE Impr: Output Heterodyne Efficiency divided by Input Heterodyne Efficiency

HE to Perfect: Output Heterodyne Efficiency divided by Perfect Heterodyne Efficiency

Strehl to Perfect: Output Strehl ratio divided by Perfect Strehl ratio

6. References

Bibliography

[GIG07] *Optical Satellite Downlinks to Optical Ground Stations and High-Altitude Platforms*, by D. Giggenbach, J. Horwath, and B. Epple, IST Mobile & Wireless Communication Summit 2007

[GIG09] *Optical Data Downlinks from Earth Observation Platforms*, by D. Giggenbach, J. Horwath, and M. Knappek, 2009 SPIE Free-Space Laser Communications Technologies XXI

[GIG10] *APHEFIS - Adaptive Optics by Phase Estimation from Focal Intensity Speckle*, by D. Giggenbach, 2010

[GI11a] *Offenlegung DE102010021340A1*, patent property of DLR, concept developed by D. Giggenbach, 2011

[GI11b] *Deriving an estimate for the Fried parameter in mobile optical transmission scenarios*, by D. Giggenbach, 2011 Applied Optics, Vol. 50, Issue 2

[GON08] *Digital Image Processing*, by Rafael C. Gonzalez and Richard E. Woods, 2008 Prentice Hall

[GOO68] *Introduction to Fourier Optics*, By Joseph W. Goodman, 1968 McGraw-Hill

[GUR97] *An Introduction to Neural Networks*, By K.Gurney, 1997 UCL Press

[HAR98] *Adaptive Optics for Astronomical Telescopes*, by J. Hardy, 1998 Oxford University Press

[KNA10] *Adaptive Optics for the Mitigation of Atmospheric Effects in Laser Satellite-To-Ground Communications*, by M. Knappek, 2010 Technische Universität München

[LAN05] *Optical inter-satellite links based on homodyne BPSK modulation: Heritage, status, and outlook*, By R. Lange, 2005 Free-Space Laser Communication Technologies XVII, Proceedings of the SPIE Vol. 5712

[PER08] *Results of the Optical Downlink Experiment KIDDO from OICETS Satellite to Optical Ground Station Oberpfaffenhofen*, By N. Perlot, M. Knapek, D. Giggenbach, J. Horwath, M. Brechtelsbauer, Y. Takayama, T. Jono, 2007 Proceedings of SPIE Vol. 6457

[ROJ96] *Neural Networks – A Systematic Introduction*, By R.Rojas, 1996 Springer

